Sensorimotor Learning for an Artificial Body Schema on Humanoid Robots

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

der Fakultät für Informatik
des Karlsruher Instituts für Technologie (KIT)

genehmigte

Dissertation

von

Stefan Ulbrich

aus Heilbronn-Neckargartach

Tag der mündlichen Prüfung: 10. Februar 2014

Erster Gutachter: Prof. Dr.-Ing. Rüdiger Dillmann

Zweiter Gutachter: Prof. Dr. Carme Torras

Deutsche Zusammenfassung

Menschen verfügen über die angeborene Fähigkeit, die Lage einzelner eigener Körperteile im Raum zu schätzen, die Gestalt ihres Körpers durch Selbstexploration zu erlernen, Bewegungen im voraus zu planen und darüber hinaus Bewegungen anderer Menschen zu erkennen und zu analysieren. Diese Fähigkeiten setzen eine mentale Repräsentation des eigenen Körpers voraus, die zugleich hochgradig anpassungsfähig als auch multimodal sein muss, um den Menschen die Möglichkeit zu geben, zu planen und dynamisch Änderungen des eigenen Körpers auszugleichen. Beobachten lassen sich diese Fähigkeiten bereits bei Kleinkindern, die direkt nach der Geburt mit der Erkundung des eigenen Körpers beginnen und sich im Laufe ihrer Entwicklung zum Erwachsenen an eine sich ständig verändernde Erscheinung anpassen, sowie in der Akzeptanz künstlicher Prothesen, die sich bei Menschen mit amputierten Gliedmaßen beobachten lässt. Die Mechanismen, die diese Fähigkeiten ermöglichen und tief im menschlichen Gehirn verankert sind, werden in den Neurowissenschaften oftmals unter dem Begriff des Körperschemas (engl. Body Schema) vereint (Head and Holmes, 1911). Die Leistung, die das Gehirn dazu erbringen muss, ist beachtlich, denn bereits an den einfachsten, alltäglichen Handlungen ist das Körperschema maßgeblich beteiligt. Insbesondere gilt dies bei der Verwendung von Werkzeugen. Es lässt sich beobachten, dass unter Verwendung solcher Werkzeuge, die den Aktionsradius vergrößern (z.B. ein Zeigestab oder auch ein Tennisschläger), der fremde Gegenstand direkt in das Körperschema integriert und somit als Teil des eigenen Körpers wahrgenommen wird. Daraus resultiert, dass dieser Gegenstand mit vergleichbarem Geschick zur Manipulation der Umwelt eingesetzt werden kann. Das Körperschema trägt somit maßgeblich zur menschlichen Anpassungsfähigkeit und der Möglichkeit mit der Umwelt auf vielfältige Weise zu interagieren bei.

Versuche, vergleichbare nützliche Eigenschaften auf künstliche Systeme wie einen Roboter zu übertragen, waren bisher wenig erfolgreich. *Humanoide Roboter*, deren Erscheinungsbild, wie der Name suggeriert, dem des Menschen nachempfunden ist, bieten eine ideale Grundlage, um die Nachbildung eines Körperschemas zu erforschen. Das liegt zum einen daran, dass ihr Körper dem Menschen ähnliche Möglichkeiten zur Interaktion mit der Umwelt bietet. Zum anderen erfordert die Regelung des Roboters genaue Modelle, die – bedingt durch den komplexen Aufbau und das Einsatzgebiet in menschenzentrierten Umgebungen – mittels klassischer Vermessung und Kalibrierung nur schwierig zu ermitteln sind. Solche Modelle sind obendrein statisch, so dass ein Roboter nicht Veränderungen seines Körpers ausgleichen oder Werkzeuge in die Repräsentierung des Körpers einbinden kann.
Gegenstand der Forschung in dieser Arbeit sind daher geeignete Modellrepräsentationen und Algorithmen, die Nachbildung eines Körperschemas auf humanoiden Robotern ermöglichen. Dabei

werden insbesondere die Arten von Modellen, die die direkte Kinematik und die inverse Dynamik des Roboters abbilden und mit den Steuer- und Sensorsignalen des Roboters verbinden, betrachtet. Das Lernen solcher Modellrepräsentationen wird als *sensomotorisches Lernen (eng. sensorimotor learning)* bezeichnet. Zusätzlich zu der Möglichkeit der Verknüpfung verschiedener Modalitäten sollen die Repräsentationen obendrein ein möglichst effizientes Lernen ermöglichen, da mit steigender Anzahl von Bewegungsfreiheitsgraden des Roboters die Anzahl der Trainingsbeispiele, die zum Lernen benötigt werden, exponentiell steigt.
In dieser Arbeit wurden viele Methoden, die sich zum sensomotorischen Lernen eignen, untersucht. Prinzipiell lässt sich jedes Verfahren zur allgemeinen Funktionsapproximation für das Erlernen der Relationen zwischen sensorischen Stimuli und Motorsignalen verwenden. Da jedoch die Anzahl von Beobachtungen während des Trainings sehr groß werden kann, wird die Wahl des Verfahrens entscheidend für den Einsatz auf einem humanoiden Roboter. Bei der Funktionsapproximation kann zwischen *globalen* und *lokalen* Lernen unterschieden werden. Modelle, die für globales Lernen geeigneten sind, ermöglichen generalisierte Vorhersagen für nicht explorierte Konfigurationen des Körpers, während beim lokalen Lernen Modelle verwendet werden, die zuverlässige Aussagen lediglich in gut erforschten Bereich des Konfigurationsraumes treffen können. Prinzipiell erscheinen somit globale Verfahren als die geeignetere Wahl. Dennoch hat sich lokales Lernen aufgrund der höheren Lerngeschwindigkeit und dem Fehlen geeigneter globaler Modelle durchgesetzt.

Aufgrund dieser Begebenheit wurden in dieser Arbeit neue Modellrepräsentationen entwickelt, die sich für die Beschreibung sowohl der direkten Kinematik als auch der inversen Dynamik verwenden lassen. Diese Modelle können mit globalen Lernverfahren ermittelt werden und zeichnen sich in Verbindung mit ebenfalls neuentwickelten Lernalgorithmen durch eine Effizienz aus, die den Stand der Forschung übertrifft. Durch ihren Ursprung im computergestützten geometrischen Design und ihr Einsatzgebiet wurden die Modelle *kinematische Bézier Karten (engl. Kinematik Bézier Maps, KBM)* beziehungsweise *dynamische Bézier Karten (engl. Dynamic Bézier Maps, DBM)* benannt. Das hervorstechendste Merkmal dieser Repräsentationen ist die Tatsache, dass die Modelle *linear* in ihren Modellparametern sind und somit die Entwicklung besonders effizienter Lernalgorithmen, die ebenfalls in dieser Arbeit vorgestellt werden, ermöglichen.
Da der Bedarf an Trainingsbeispielen während des Lernens jedoch exponentiell bleibt, wurde im Rahmen dieser Arbeit ebenfalls ein Algorithmus entwickelt, der im Falle der direkten Kinematik eine Reduktion der Komplexität zu Lasten einer komplizierteren Explorationsstrategie ermöglicht. Basierend auf bereits vorhandenen Dekompositionstechniken, die das Lernproblem in einfachere Teilprobleme zerlegen, wurde ein neues Verfahren entworfen, das weder Vorraussetzungen an die Struktur des Roboters stellt, und somit allgemeiner ist, noch erhöhte Anforderungen an die Sensorik des Roboters stellt. In Kombination mit diesem Verfahren ist es möglich, zusammen mit den KBM die kinematische Struktur auch sehr komplexer Roboter mit vielen Freiheitsgraden zu erlernen.

Alle in dieser Arbeit entwickelten und vorgestellten Verfahren wurden ausgiebig evaluiert und mit dem Stand der Forschung verglichen. Dazu wurden sowohl Experimente in Simulation durchge-

führt, um die generellen Eigenschaften der Methoden zu untersuchen und zu belegen, wie auch Experimente auf dem humanoiden Roboter ARMAR-III durchgeführt, die die Anwendbarkeit auf einem echten System verdeutlichen.

Contents

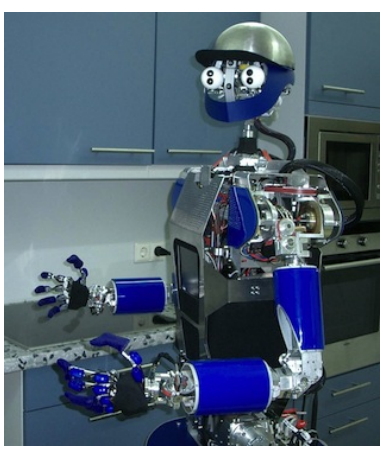
1. Introduction

1.1. Motivation

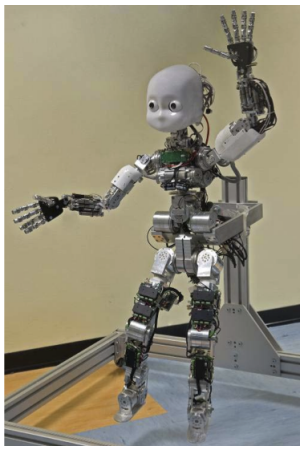
The body schema Human beings possess a natural unconscious awareness which enables them to learn and explore their bodies from pure self-observation, to plan motions, and even to recognize and analyze the movements of others individuals. The mental representation of the own body is highly adaptable and allows us humans to dynamically compensate for changes of our body. This amazing phenomenon can be observed in newborns that already begin to explore their bodies, in the adaptation to growth during human development or in how amputees manage to handle their prosthesis. The mental representation of the own body within the cerebral system is called the *body schema* (Head and Holmes, 1911).
Even in the most trivial motoric day-to-day actions, the body schema is involved. In particular, this includes handling tools. In such cases where the desired effect of tool use is the extension of the own body and its action space (when using a rake for instance), the tool literally becomes incorporated into the body schema ("makes it disappear from the immediate awareness"[1]). This phenomenon grants an agility similar to as if the tools were a real part of the body. The remarkable human ability to acquire and maintain a coherent body schema leads to a high degree of adaptability and autonomy and facilitates sophisticated interaction with the environment. To date, a comparable cognitive skill could not successfully be recreated in artificial systems such as robots.

Humanoid robots As the name suggests, *humanoid robots* are directly modeled after the human body and consequently possess an embodiment that intentionally has a great resemblance to the human appearance (see Figure 1.1). It is especially this shape that makes these robots appear 'human' and raises the expectation of human-like behavior and cognition. It also opens the possibility to move and interact in our environment the same way as humans do. However, this versatility comes at the price of high complexity. The control of many degrees of freedom (DoF) implied by the humanoid shape is difficult to handle. For instance, calculating the joint configuration to a given shape (*the inverse kinematics*) becomes a challenging task with infinitely many solutions as human limbs are composed of redundant joints.
In addition to that, classical calibration of such mechanisms can be extremely expensive and time-consuming requiring sophisticated instruments. Today, a robot still cannot perform a calibration all by itself as it lacks the necessary adaptability. Relying on classical techniques leaves it unaware of a possibly continuously changing body structure and incapable of adaptation—a basic property that humans rely on already from on the very beginning of their early development. Even though the body of a humanoid robot does not develop,

[1] Nabeshima et al. (2007)

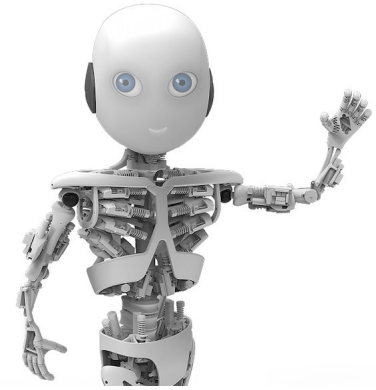


(a) ARMAR-III. *Source: (Asfour et al., 2006)*



(b) iCub.
Source: (Metta et al., 2008)

(c) HRP-4C.
Source: (Kaneko et al., 2009)



(d) Roboy.
Source: awtec, 2012

(e) ARMAR-4. *Source: (Asfour et al., 2013)*

Figure 1.1.: Assortment of humanoid robots.

it undergoes many changes throughout its lifetime. This often occurs after repair or may be the result of wear and tear in tendon driven joints for instance. Another example is handling a tool—the dynamic connection between a complex hand and arbitrary tools can neither be predicted during the design stage nor be planned beforehand.

Artificial body schema The examples above motivate the development of cognitive mechanisms which eventually will manifest in an artificial body schema. where machine learning is used to acquire the necessary sensorimotor skills. This is a challenging task, however, and the learning problem becomes critically complex when confronted with the high number of independent degrees of freedom that a humanoid robot typically possesses. The development and evaluation of suitable and efficient methods for learning and dimensionality reduction is an essential step towards the establishment of an artificial body schema for greater autonomy of humanoid robots.

1.2. Problem statement

The biological body schema is multimodal which means that it relates multiple *sensor* and *motor control* signals to each other. The different sensor modalities can be hereby divided into two categories: *i.) Exteroception*, the perception of the outside world, reacts to stimuli outside the body (for instance, vision, touch, audition and forces). *ii.) Proprioception* denotes the perception of the own body state, for instance muscle tension or joint positions on a robot. As such, they are closer related to the motor control signals. Attempts to recreate the body schema on robots by means of machine learning mostly consist of constructing mappings between the high-dimensional spaces of these signals—the so called *sensorimotor maps*. The creation of sensorimotor maps usually refers to supervised and continuous function regression that learns from observations in form of proprioceptive input and exteroceptive output values. An example of a sensorimotor relation is the *forward kinematics*—the most basic and most important aspect of the body schema. This is the relation between the visually perceived state of the body, the position of the end-effector for instance, and the respective motor control signals or joint angles as perceived by the proprioception.
In contrast to classical approaches, which either involve extensive measurement of the robot's kinematics and dynamics or rely on possibly inaccurate or distorted models from computer aided design (CAD), learning means a more flexible alternative that does not require manual modeling and therefore grants a much higher degree of autonomy. However, the achievable accuracy of the models depends on the exploration and the associated quantity of observable training samples such that there is always a tradeoff between accuracy and autonomy: In case of an industrial robot, accuracy is the highest priority. In contrast, a humanoid robot's purpose is the autonomous interaction in a human-centered environment where precision is less important.

In principle, many nonlinear function regression algorithms are potentially well suited to learn nonlinear relations between sensor stimuli and motor signals. However, the number of observations necessary to learn an accurate representation is often exponential in the number of sequential degrees of freedom. The right choice of learning algorithm then becomes crucial: *Local regression* quickly produce accurate local representations but generalizes poorly in unexplored regions while *global regression* and *model-based approaches* generalize well but are notoriously slower. The ability to incrementally update the sensorimotor map is also a necessity for meeting the requirement of online adaptability, that is, to learn during application.

1.3. Contributions

This work contributes to the implementation of artificial body schemas on humanoid robots that replace classical modeling, calibration, and system identification methods. Therefore, novel computational models are proposed and evaluated that are especially designed for sensorimotor maps, and suitable learning algorithms are adapted and newly developed that exceed current state-of-the-art techniques with respect to precision and learning from few examples. In addition, an effective novel approach for dimensionality reduction is proposed. The main contributions of this thesis are summarized as follows:

- **Representations for kinematic and dynamic models**
  The main contribution of this work are novel methods for the representation of kinematics and dynamics such that their model parameters can be easily determined.

  i.) The *Kinematic Bézier Maps (KBM)* are models for machine learning created in this thesis. They are derived from projective geometry and have an inductive bias that specializes them to combinations of trigonometric functions. This makes them predestined as models for learning the forward kinematics (FK) of serial robot manipulators with only rotational joints which includes humanoid robots. Learning creates a mapping from the configuration space to a manifold that describes all possible end-effector positions. The most prominent property of the KBM is that they represent this mapping as a linear combination of spatial points (that is, the model parameters). Experiments with a humanoid robot show that machine learning using the KBM perform better than the state-of-the-art. They require fewer examples for the same precision and predict values even for unexplored regions.

  ii.) In addition, the concept of the KBM is transferred to learning the inverse dynamics of robotic systems. Starting from the Lagrangian formulation of the equations of motion, the KBM are modified to represent the components that influence the joint torques necessary to achieve or maintain a given state of the dynamic system. The model has a significantly higher number of parameters and learning the inverse dynamics is slower compared to learning the FK but still benefits from the linearization of its target function.

- **Learning kinematic and dynamic models**
  The most prominent property of the KBM is the linearization of complex sensorimotor relations and it facilitates the adaptation and creation of very efficient learning algorithms:

  i.) *Linear least squares methods* are adapted to build models from given data sets in a single step (*Batch learning*). They are capable of building exact models if there is no sensor noise. For instance, in simulation, these models are exact representations of the sensorimotor relation. However, some of them also are sensible to sensor noise which leads to overfitting which has to be compensated by redundant training samples. Therefore, a more sophisticated regression technique has been adapted for learning with the KBM.

  ii.) *Incremental learning* applies the δ-rule usually known for learning single-layer artificial neural networks to a previously learned model. This results in instant and mainly local adaptation to alterations of the body but learn much slower compared to batch learning. However, they are less prone to overfitting and allow the adaptation of initial models obtained from simulation on real hardware.

  iii.) *nonlinear symmetry optimization* is a more complex and novel algorithm that enforces symmetry constraints of the KBM models during learning which greatly reduces the risk of overfitting to adverse training data.

- **Dimensionality reduction through decomposition**

  Independently of the chosen machine learning method, the problem of sensorimotor learning remains exponential. The complexity, however, can be influenced by an exploration strategy—at least so in the case of the forward kinematics. This strategy implements a novel *decomposition* technique developed in this thesis that, unlike previous work, neither restricts the kinematics nor requires additional sensors. The kinematics is decomposed into a *lateral* and a *proximal* partial kinematics preferably of the same length which have to be successively learned. This reduces the dimension of input spaces of both partial kinematics and reduces of the number of required training samples to its square root rendering learning more applicable.

1.4. Outline

This thesis documents the contributions to sensorimotor learning which is essential for the implementation of an artificial body schema on humanoid robots. Therefore, the second chapter gives an overview over the related work including neuro-scientific research, developmental robotics and tool use in robotics and presents the state-of-the-art in sensorimotor learning. The next chapters present the contributions to this field in detail—the novel models for sensorimotor learning in chapter three, the associated learning algorithms in chapter four and, in chapter five, the approach for dimensionality reduction. In chapter six, the concepts and results of the evaluation both in simulation and with experiments on a real humanoid platform are documented. The work concludes in chapter seven with a discussion and summary of the presented methods and their evaluation and an outlook on future work.

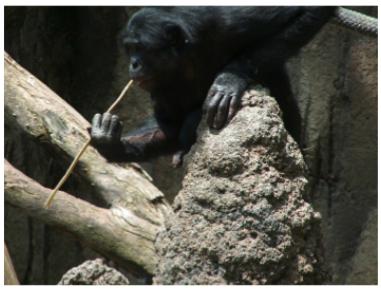
2. Related Work

In this chapter, all work related to the creation of an artificial body schema for robots as well as closely related topics will be presented and discussed. In the first section, an overview over related research in neurosciences is be given. Afterwards, a history of conceptual work of modeling a body schema for robots on a higher abstraction level is given. This refers to the work that does not directly focus on modeling the mechanisms in the brain that relate high-dimensional sensor to motor signals but rather on the design of complete architectures or cross-modality relations. The topic of tool use in robotics is also addressed in this section as well as a summary on traditional approaches preceding the idea of body schema learning such as modeling, calibration and parameter estimation. Section three exhibits the state of the art in sensorimotor learning. As this thesis concentrates on the development of novel sensorimotor learning algorithms, this section is the most important and therefore most comprehensive in this chapter. In there, all relevant methods are analyzed and compared and the relevant design decisions affecting the development of this thesis are discussed. It provides the fundamental terms and ideas and therefore, it contributes heavily to the understanding of the methods presented in subsequent chapters.

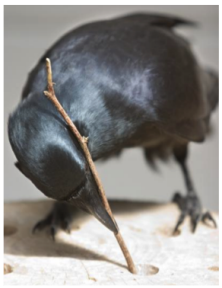
2.1. The body schema in neuroscience

The execution of coordinated and complex actions requires an awareness of the current state of the body that humans naturally possess. It allows us to interact with our environment and open doors, for instance, even without the need for visual feedback. In this example, we can reliably estimate the pose of our hand and other body parts. Further, this ability has to adapt to a constantly changing embodiment, for instance during human development and after injury or impairment. It is most remarkable that, when handling a tool that extends the reach, it disappears from the immediate awareness (Nabeshima et al., 2007) after a short period time and can be handled with the same dexterity as if it were one with the body. Although human capabilities in tool use are unmatched, this ability is not unique to humans only. They can be observed in many animals obviously ranging from primates to animals not particularly known for such highly developed cognitive abilities such as birds and even invertebrates (for a review, see Shumaker et al., 2011) (see Figure 2.1).

In the human brain, the awareness of the body requires dynamic representations of the embodiment that contain information on the hierarchical arrangement of body parts (body topology) and their relative positions. They also have to integrate feedback from many different senses and motor commands into a coherent mental image of the body (Haggard and Wolpert, 2005). These neural representations and their nature have been the subject of research in neuroscience already for a long time. Head and Holmes (1911) first introduced the concept of *body schema* for the action-related and unconscious awareness of the body, while the



(a) Chimpanzee. *(Credit: Mike Richey)*



(b) Corvus moneduloides. *(Credit: Dr. Simon Walker)*

(c) Veined octopus. *(Credit: Nick Hobgood)*

Figure 2.1.: Tool use in the animal kingdom.

more conscious and mainly visual perception but also emotional perception of the own appearance is often referred to as *body image* (Paillard, 1999; de Vignemont, 2010). However, there exists no clear definition of the body schema and its interpretations vary throughout different disciplines. It is out of the scope of this thesis to report the comprehensive scientific literature on the body schema in detail and the interested reader is referred to the reviews in (Berlucchi and Aglioti, 2009; Maravita and Iriki, 2004; Maravita et al., 2003). This section focuses on the properties of human body awareness and selected evidence for its existence, and adopts the characterization proposed by Haggard and Wolpert (2005) (who preferred the term *body scheme* to prevent ambiguity). The characterization is based on properties supported by evidence observable in lesion patients exposing a dysfunctional body schema (after a stroke for instance) and experimental studies with humans and monkeys.

Spatial encoding: First, the body schema obviously is *spatially encoded* as it grants an awareness of the position of the body parts in the external 3D space. Therefore, it integrates tactile and proprioceptive sensations (Head and Holmes, 1911) and makes stimuli on the body surface localizable in the external space. Gallagher and Cole (1995) reported the case of a patient with peripheral deafferentation, that is, the loss of proprioceptive and tactile sensation below the neck. Although the patient could partially recover control of its body when relying heavily on visual feedback, even blind people possess better sensorimotor control emphasizing the importance of the tactile and proprioceptive senses.

Modularity: Another important property is the *modularity* of the body schema. Stamenov (2005) suggested that the body schema is not a single wellformed pattern but rather a set of several connected groups of neurons that represent opportunistically learned manifolds that are distributed over different regions in the brain. There also is evidence for the encoding of individual body parts in different neural modules (Tessari and Rumiati, 2002). Disorders in the body schema that affect only the fingers (finger agnosia in this case, the inability to name and identify the own fingers) indicate the existence of a separate 'finger schema' (Benton, 1959). Surprisingly, the simultaneous occurrence of finger agnosia, dyscalculia (impairment of understanding arithmetics), left-right confusion and agraphia (the inability to write despite mental and sensorimotor ability) in the *Gertmann syndrome* (Gerstmann, 1942) is interpreted as evidence for a relation between the modularity of the body schema and abstract and symbolic thinking in general (Haggard and Wolpert, 2005).

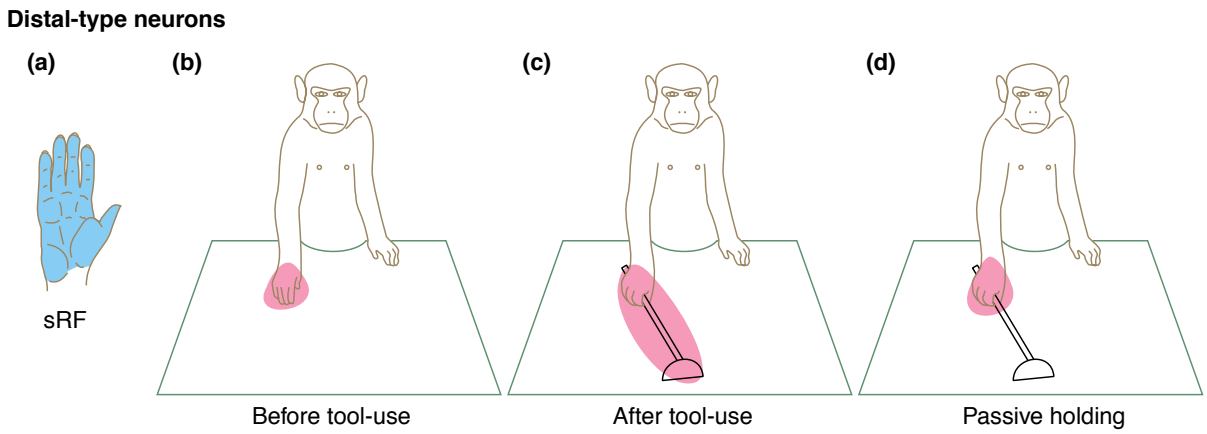


Figure 2.2.: *Source: (Maravita and Iriki, 2004)*, TRENDS in cognitive science

Update by movement: The brain updates the body schema when the body moves (it is "updated by movement" (Haggard and Wolpert, 2005)) to maintain a representation that is always synchronous and consistent with the current body state. Experiments supporting this property focus on special *bi-modal neurons* that respond to proprioceptive information and visual stimuli localized within a *receptive field* (a volume in the external space). Graziano and Gross (1993) showed that many bi-modal neurons in the parietal lobe of the brain, which respond to visual stimuli close to one hand, update the location of the receptive fields to follow the hand's movement.

Adaptability: The most notable property of the body schema is its *adaptability*. As stated earlier, the body schema gradually adapts to a growing body or incorporates tools in shorter time spans. Iriki et al. (1996, 2001) showed that, during tool use, the receptive fields of bi-modal neurons in the brain of Japanese macaque monkeys previously linked to the position of the hand (peripersonal space) extended towards the tip of the tool. A similar experiment was reported in (Maravita and Iriki, 2004) that examined the distal visual receptive field (the area in external space where stimuli activate the neurons associated to grasping). It was shown that this field was enlarged by the length of a tool that the monkeys used once they had been trained to do so (see Figure 2.2). These experiments lead to the theory that the tool itself became incorporated into the monkey's own body schema. Similar conclusions were drawn from experiments with human patients who suffered from brain damage or phantom pain after having lost a limb. Although the methodology of the ground-breaking work of Iriki et al. (1996) has been criticized by Holmes (2012), recent research confirms the plasticity of the body schema and the peripersonal space (Canzoneri et al., 2013; Holmes, 2012) The observation that the effects of tool use can still be observed after the tool has been laid aside (Cardinali et al., 2012) further supports this theory. Maravita and Iriki (2004) provide an extensive review of the research on the incorporation of tools into the body schema. The brain regions that are active during tool use can be detected and localized by functional magnetic resonance imaging (fMRI) and position emission tomography (PET). For a review, the reader is referred to (Lewis, 2006).

Apart from tool use, the plasticity could be observed in an experiment by Botvinick and Cohen (1998)—the famous *rubber hand illusion*. Participants receive tactile stimuli on one hand which is hidden from their vision. At the same time, they observe the same stimulus applied to a rubber

replica close to the position of the real hand. After a short time, the participants had the illusionary sensation to feel the stimuli on the rubber hand instead of on their own. They established a *sense of ownership*. This impression is so strong that the participants experienced fear when they observed a syringe close to the rubber hand (Armel and Ramachandran, 2003).
The limitations of the adaptability were investigated in an interesting experiment by Kilteni et al. (2012). In virtual reality, the participants experienced a different embodiment with elongated arms controlled by their real movements. They showed that the body schema easily adapted to virtual limbs up to a length of three times of the length of the real body parts.

Supra-modality: In order to maintain a spatial representation of the current body state, the body schema has to process various (and possibly redundant) sources of information from multiple senses such as proprioception, hearing, vision and the somatosensory system (touch). This property is called *supra-modality* or *multisensory* (Heed and Röder, 2012; Maravita et al., 2003; Lackner, 1988). For instance, Rorden et al. (1999) showed with the rubber hand illusion that simultaneous visual and tactile stimuli on the same body part amplify each other and hence are integrated into a single representation.

Coherence: While processing information from different sources with different biases for the integration into a representation of the body, the brain optimally resolves discrepancies in these signals to obtain a *coherent* body representation (van Beers et al., 2002). Lackner (1988) describes an experiment that shows the capability of the brain to actively compensate discrepancies in the perception: A blindfolded person had to touch his nose while his proprioception was actively distorted by mechanical vibrations on the biceps tendon creating the illusion of an extension of the forearm. The subject's brain compensated the discrepancy by altering the perceived size of the subject's nose to conserve the information of the contact between the finger and the nose. Reversely, the opposite effect can be generated by vibrations acting on the oponent tendons (Longo et al., 2009).
Ritchie and Carlson (2013) investigated *afterimages* during tool use. First participants of this experiment had to adapt to darkness. A single flash of light then creates a long-lasting visual impression, the so-called afterimages. When a limb is moved, it fades away much more quickly than the rest of the image. This confirms the idea of a coherent body schema where the brain actively resolves the discrepancy in the afterimage and the proprioception by manipulating the perceived image. This experiment also supports the theory of tool incorporation as the tool disappears together with the limb from the afterimage.

Interpersonality: Finally, the body schema is not only the representation of our own perceived body but also enables us to recognize the body state of *other individuals (interpersonality)* and to analyze their actions through observation. Experiments by Tessari and Rumiati (2002) and Reed and Farah (1995) confirmed this theory by showing that observed actions can be better analyzed and recalled by simultaneous imitation. The *mirror neural system (MNS)* are groups of neurons related to this phenomenon. Their activity can not only be detected during motor actions but when these actions are observed by other individuals. The effect is the strongest for actions that involve grasping and mirror neurons seem to be strongly related to the *canonical neurons* which response to the recognition of objects that require a specific grasp type (see Lopes and Santos-Victor, 2005; Murata et al., 1997).

Stamenov (2005) proposed that the mirror neuron system is mainly responsible for the creation and the interlinking of the modular body schema in the brain.

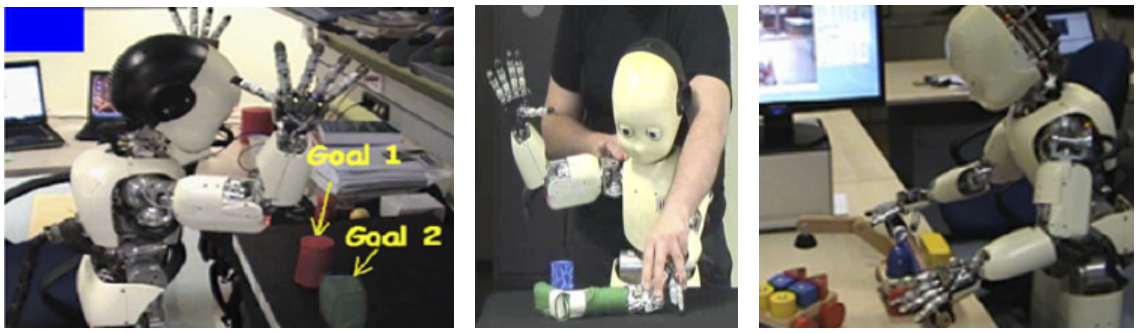
In sum, the body schema is a modular, coherent, adaptable awareness of our spatial configuration of the body and that of other observed individuals. The development of an artificial body schema for humanoid robots should therefore be driven by the reproduction as many of these characteristics as possible.

2.2. The body schema in robotics

The last section gave an impression of the cognitive abilities of humans to create and maintain action-oriented representations of the own body and the resulting cognition that inspired researchers to try to reproduce similar abilities on artificial systems. Over the last years, the development of *artificial body schemas* has gained a lot of attention and led to its own field of research in robotics. This section reviews the evolution and progress of related studies divided into traditional approaches such as calibration and parameter estimation, the conceptual design of cognitive architectures, developmental robotics, and the closely related tool use in robotics. These subjects were also addressed in a very comprehensive survey by Hoffmann et al. (2010) which is an excellent source of further information.

Traditional approaches At first, classical approaches of robot modeling, calibration and parameter estimation should be briefly mentioned here as they were the predominant methods before the emergence of biologically inspired approaches and still are predominant in industrial applications and represent an enormous field of research for themselves. As a predecessors to adaptive body representations, models of the kinematics and dynamics of robot manipulators had to be created by experts that required detailed knowledge on the structure and properties of the robots (Sciavicco and Siciliano, 2000; Craig, 2005). Because such models are expressed as algebraic functions that directly reflect this knowledge, they are called *explicit* (Hoffmann et al., 2010). After the process of manufacture, the parameters of the models have to be calibrated to compensate for production tolerances. The creation of accurate explicit models is expensive with respect to manual work and measuring equipment but enable the precise control in the complete workspace of the robot without the need for external perception. The biggest disadvantage, however, is that they are static, that is, the robot has to be calibrated after every structural change (for instance, caused by wear and tear). If explicit models are parameterized, the process of calibration can be made more flexible by *parameter estimation* (Ljung, 2009) and *autonomous self calibration* (Bennett et al., 1991; Ma and Hollerbach, 1996; Hollerbach and Wampler, 1996). This requires at least an additional source of perception (a camera for instance) and reduces the cost as human intervention can be minimized. That way, even tools can be integrated into the body representation (Hersch et al., 2008; Martinez-Cantin et al., 2010)
When robot systems grow more complex—especially in the case of humanoid robots—, explicit models become intractable and the focus shifts from precision to flexibility. Biologically inspired methods such as artificial neural networks (see Section 2.3.3) gain importance as explicit models are more difficult to obtain.

The design parameters cannot explicitly be derived from such representations which is why they are called *implicit models*.

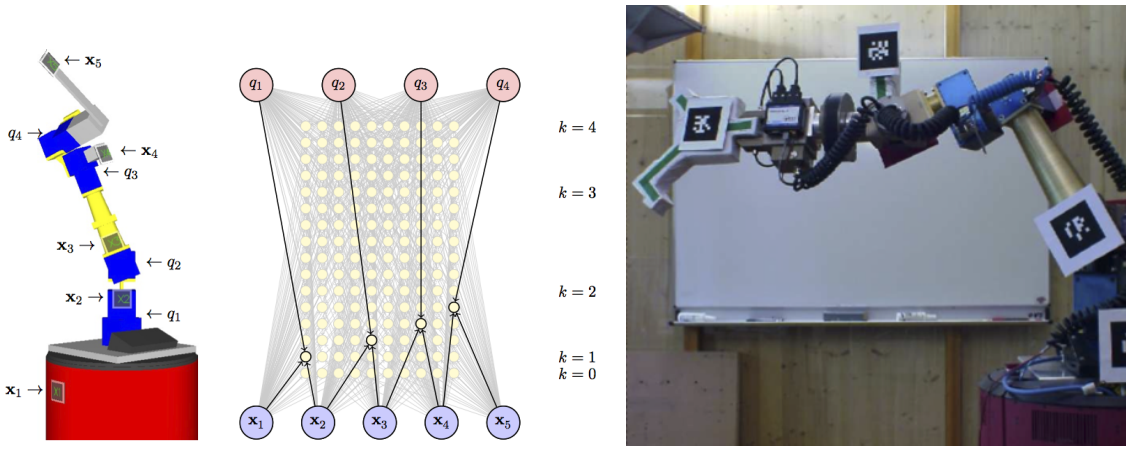


(a) Bi-manual reaching. (b) Tool use. (c) Using a toy crane.

Figure 2.3.: The humanoid robot iCub learning complex tasks. *Source: (Morasso, 2013)*

Conceptual artificial body schemas One of the first to use the notion of the 'body schema' in robotics were Morasso and Sanguineti (1995). They presented a non-parametric statistical technique closely related to the self organizing feature maps (Morasso and Sanguineti, 1995) later presented in Section 2.3.3. For control, the model could be combined with a potential field to solve the inverse kinematics. It was extended by Stoytchev (2003) to be extendable for the incorporation of tools by representing the tooltip in the coordinate frame of the end-effector. Very recently, Morasso (2013) and De Santis et al. (2013) asked the questions "Do humanoid robots need a body schema" and "What is the use of the body schema for humanoid robots?". Less complex mobile robots (for instance, the Braitenberg vehicles (Braitenberg, 1986)) exploit their morphology to express complex, coordinated and adaptive behavior by purely reactive neural circuits (Morasso, 2013). They consequently do not require a complex representation of their bodies. Humanoid robots, on the other hand, have many redundant degrees of freedom that are much more difficult to control and their universal embodiment requires a body schema for the same reasons as the human body does in their opinion. In their work, they interpreted the body schema as a middle-ware between motor control and cognition. Goal-oriented actions that involve movement ('overt actions') rely as much on a model of the body as 'covert actions', that is, actions that are not executed on the hardware and serve as simulations for high-level planning.

Over the years, their research evolved into a comprehensive computational model for an artificial body schema (Mohan et al., 2013; Mohan and Morasso, 2012; Mohan et al., 2011a,b, 2009; Mohan and Morasso, 2007) on the humanoid robot iCub (see Figure 1.1b). Their approach heavily relied on force-fields ('passive motion paradigm' (Mussa Ivaldi et al., 1988) formalism) for the control in accordance with the *equilibrium point hypothesis* (Feldman, 1966) that states that motion is controlled such that an equilibrium between all muscular and environmental forces is maintained. That way, they could perform complex bimanual and full-body actions even for complex tasks (see Figure 2.3) such as tool use. The transition between planning in external space and execution in the configuration space is realized by first-order Taylor approximations (Jacobians) that can be obtained from general function approximation by neural networks such as multilayer perceptrons (see Section 2.3.3).



(a) (b) (c)

Figure 2.4.: A probabilistic framework for learning an artificial body schema. *Source: (Sturm et al., 2012)*

Cabido-Lopes and Santos-Victor (2003) presented a complete architecture for the imitation and recognition of gestures without a deeper understanding of their meaning. Their work was motivated by the idea that traditional programming of humanoid robots is very inefficient and that the natural and intuitive interaction should be a long-term goal in robotic research. The discovery of mirror neurons (see Section 2.1) used both for the body schema and the recognition of body postures of other individuals drove the development of sensorimotor maps for an anthropomorphic robot arm with four degrees of freedom. Therefore, they used artificial neural networks (*multilayer perception*) and decomposed the learning into a sequence of smaller learning problems (a similar concept as presented in Section 2.3.6) motivated by the increasing abilities during human development (Metta et al., 2000). After learning in a phase of self-observation the sensorimotor map could be used in combination with a viewport transformation to detect gestures shown by a human teacher. Later work (Lopes and Santos-Victor, 2005) added context information about objects motivated by canonical neurons (Murata et al., 1997) in the brain and iconic visual representations of hand postures. That way, their recognition method was extended to detect action goals by the application of a Bayesian network for classification.

Sturm et al. (2012, 2009) presented a probabilistic framework for learning a body schema of robots that learned geometric relations between the observed poses (position and orientation) of a robot's segments in dependance of motor control signals (so-called 'local models') as well as learned the robot kinematic topology, that is the arrangement and the type of its joints and segments. It allowed prediction and control of the robot even in the presence of hardware failures. The structure of the robot is transformed into a fully connected Bayesian network (Pearl, 1988) and local models were learned with Gaussian process regression (GPR, see Section 2.3.5) for every possible combination of motor signals. As GPR is also a probabilistic approach, it could be easily integrated into the optimization of the Bayesian network. The detection of the optimal network topology boiled down to the search of a minimal spanning tree that discards local models that were either too inaccurate (depending on the wrong motor signals) or too complex (depending on irrelevant motor signals). This is depicted in Figure 2.4b for an industrial manipulator with four degrees of freedom (Figure 2.4a). Local signals could depend on multiple motor control signals (spherical joints for instance) but learning suffered from the curse of dimensionality for higher numbers.

Figure 2.5.: Example of the 'subjective body schema' in (Hersch et al., 2008). On the left, the real ('objective') body of the robot is shown while two images on the right show possible ('subjective') valid variants of the same robot. *Source: (Hersch et al., 2008)*

The framework relied on multiple visual markers attached to the surface the robot's segments (Figure 2.4c) such that their pose could be detected by an external monocular camera (Fiala, 2005). Because of the redundancy induced by the many markers, this approach scaled well to many degrees of freedom (in here, seven) and bears a great resemblance to the work of Ruiz de Angulo and Torras (2008). Although, the markers did not have to be visible at all times and it was possible that more than one degree of freedom lied in between them, this is still a requirement that cannot be met on most humanoid platforms: Learning from self-observation on humanoid robots means relying on visual perception of the own body by stereo cameras mounted in the head such that pose detection in general and on more than one segment is often not viable.

Hersch et al. (2008) took a different approach. Unlike those of Sturm et al. (2012), their biologically motivated body schemas for humanoid robots required the kinematic topology to be static and known in advance, that is, the number of joints and in which order they are connected. On the other hand, the body representation could be learned by self-observation as only the *end-effector position* had to be observable during training. Unlike implicit models found in classical machine learning, the method applied here obtained the explicit 6D parameters of the rigid body transformations between all body parts in an gradient-based optimization process—the *iterative estimator of rigid body transformations (IERBT)* (Hersch et al., 2011). The authors stated that this is conform to biological findings suggesting that the sensor information is partially coded in frames of reference lying in the body parts (Graziano and Gross, 1993) and would enable the integration of tactile information into the same representation in future work. There is an infinite number of possible rigid body transformations that explain the training data and it could be determined to which solution the optimization would converge. The authors referred to this as a 'subjective body schema' (see Figure 2.5). Further, the IERBT algorithm is not guaranteed to always converge to a valid solution for more than one degree of freedom. Above all, convergence was very slow and required numbers of training samples in the order of $1 \cdot 10^6$. This rendered its application on a full humanoid robot inapplicable as the exploration would require days to collect the millions of necessary samples according to the authors. As shown later, implicit models as used in machine learning can find such solutions much faster but cannot construct the intermediate frames of references. To speed up learning, the authors suggested the application of intermediate markers on the robot arms that would also reduce the ambiguity of possible solutions. This suggestion is very similar to the work of Ruiz de Angulo and Torras (2008).

The work of Martinez-Cantin et al. (2010) improved the idea of Hersch et al. by using a more sophisticated optimization method—the *recursive least squares (RLS)* estimator—to calculate the rigid body transformations between consecutive joints. Moreover and in contrast to exploration by motor babbling (random movements), they presented a probabilistic *active learning* strategy for the selection of joint configurations during exploration based on the potential gain of information. In a Bayesian approach, the probability density of a model parameter given an observed position and respective joint angles was linearized and the model parameters were updated by the RLS accordingly. The constantly refining model could then be used to decide the direction of movement during exploration. They compared the results from experiments on a humanoid platform with the approach of Hersch et al. (2008) and showed that RLS in combination with active learning clearly outperforms the gradient-based optimization. Yet, both methods rely on knowing the arrangement of the joints and require significantly more training data than learning algorithms with implicit models (see Section 2.3).

Yoshikawa et al. (2002) had a completely different interpretation of an artificial body schema. They did not focus on learning the forward model of the kinematics or the dynamics but rather on learning the causal relationship between multiple sensor modalities. This was motivated by the apparent observation that if the robot touches itself, the visual observable distance between the involved body parts had to be very small. Following the principles in (Asada et al., 2001), the robot had to acquire all necessary kinematics and spatial properties from self-observation without any prior knowledge. Therefore, they combine tactile sensations, proprioception and vision into a *'cross-modal map'* as isolated senses alone cannot be interpreted without prior knowledge of the body structure because of the missing frames of reference. On their robot, they attached tactile sensors to the limbs and the hand that provided binary contact information on contacts and the position of the body parts were localized by stereo cameras in the robot's head. The cross-modal map itself was a neural layer where each neuron represented a sensor signal which was quantized in the case of vision and proprioception. The network was trained following the Hebbian rule (Hebb, 1949) such that simultaneous stimulation created a causal relationship, that is, "Cells that fire together, wire together.", (Doidge, 2007). The body schema that resulted from this procedure was suitable for the prevention of self-collisions but less so for control due to the quantization error. In (Yoshikawa et al., 2004), the problem of body extraction in uninterpreted visual data with the help of proprioceptive data addressed (that is, an artificial *body image*). Their most recent work (Yoshikawa et al., 2007) presented a framework that enables building cross-modal maps while exploring the whole body. The main problem they addressed rises from the (multiple) self-occlusion that aggravated the detection of unique causalities between touch and vision. To find pairs of correlated sensations, they proposed a neural structure with two layers—one for tactile and one for visual sensations—and a learning scheme based on the Hebbian learn rule called *cross anchoring* (see Figure 2.6). Each neuron in the tactile layer represented a discrete arm configuration whereas each neuron in the visual layer referred to a configuration of the head joints. This means that the visual space was encoded in the joint configuration in the head and the hand-eye kinematics was decomposed (see Section 2.3.6). Connections between neurons of the two layers representing a kinematic posture were then connected and reinforced when a tactile stimulus occurred and a synchronized self-occlusion had been detected. The increase in the connection weight, however, depended on the connectivity of the neuron and was damped by the sum of the connections to other neurons. The connections of neighboring neurons were weakened (called *lateral inhibition*) such that unique occurrences were learned much more quickly. Their approach

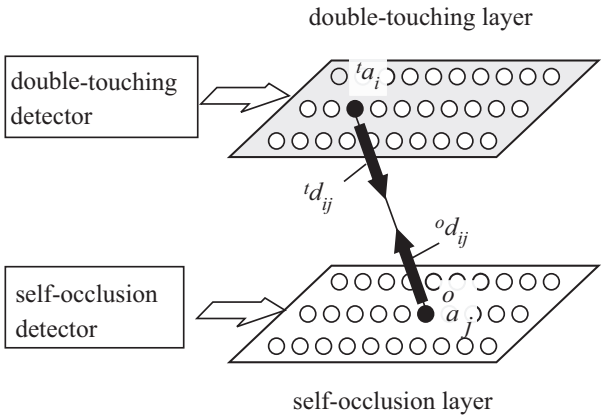
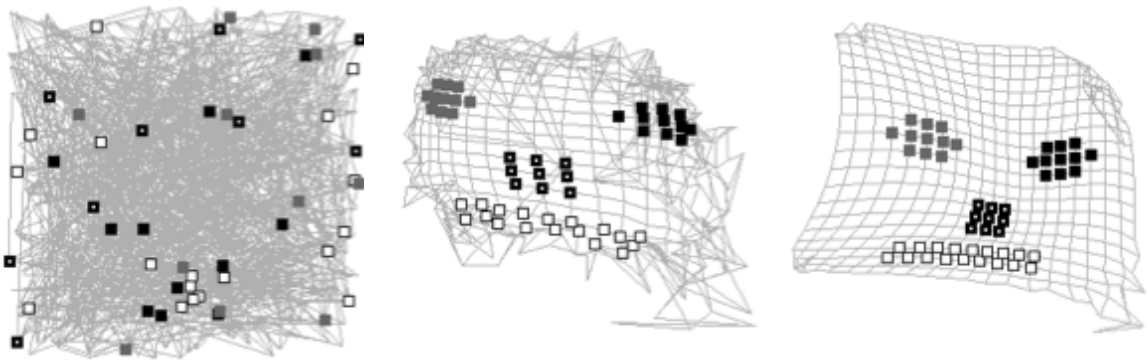


Figure 2.6.: Architecture of the cross-anchored learning. *Source: (Yoshikawa et al., 2007)*

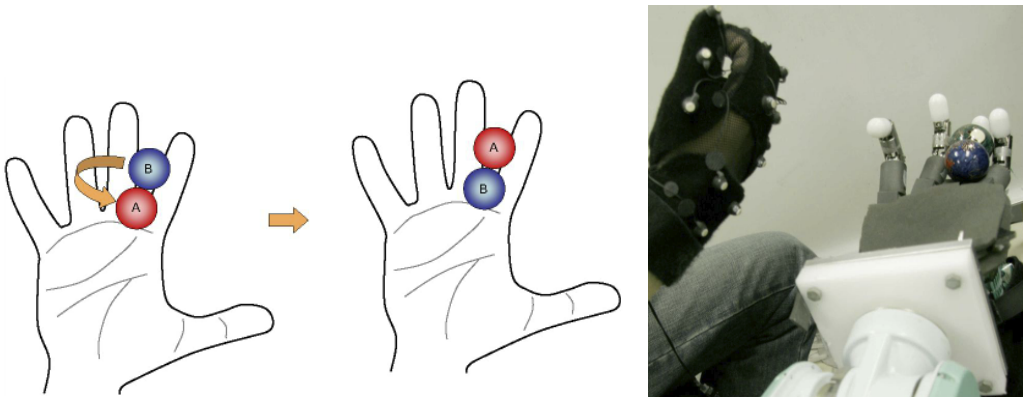


(a) Starting point. (b) After 2400 steps. (c) After 7200 steps.

Figure 2.7.: A framework for exploring invisible regions on the body. *Source: (Fuke et al., 2007)*

is intriguing because of the integration of many different modalities and is surprisingly straightforward. However, It required constant visual feedback and could not be applied to explore regions of the robot surface that cannot be observed by the camera such as the face and the back. This problem was addressed by Fuke et al. (2007) who developed a framework for hand-to-face cross-modal mapping. In a first phase, a neural network learned a sensorimotor mapping from motor control signals to the optical flow (Lucas and Kanade, 1981) induced by the movement of the end-effector when the hand was directly in front of the face. For the small angular changes required to touch the face, this sensorimotor relations were assumed to be a sufficient approximation for the exploration. When the hand pose could be estimated, the robot began to explore its face and created a cross-modal map that includes the tactile sensors (learned as a *self-organizing map*, see Section 2.3.3), the estimated hand pose and the proprioception (see Figure 2.7). The exploration of the back or the robot as supposed for future work, however, would require a more sophisticated sensor-motor mapping. In (Hikita et al., 2008), the concept of learning cross-modal maps with the Hebbian learn rule (Hebb, 1949) was combined with visual attention by saliency that allowed the robot to learn visual receptive fields of its hands or a tool.

An interesting case is the work of Oztop et al. (2006). Instead of learning an artificial body schema of their robotic hand with sixteen degrees of freedom, they exploited the adaptability of the human cerebral body



(a) (b)

Figure 2.8.: Teaching of a ball-swapping motion. The teacher exploits its own body adaptation to demonstrate the correct movements. *Source: (Oztop et al., 2006)*

representation. In their experiments, the movements of the participants' hands were optically tracked and mapped onto the robotic hand. After a learning period that took several days (mostly because of the lack of tactile feedback), the participants could control the robot hand dexterously enough to perform ball-swapping movements confirming the plasticity of the human body schema. The authors suggested, that their approach could also be applied to teach a robot how to walk like a human.

Robot tool use So far, the presented related work relates to learning a plastic body representation on robots and did not directly address tool use. The topic of tool use, however, is perhaps the most intriguing aspect of the body schema and enables humans to "move beyond the limits of their bodies" (Nabeshima et al., 2006). The following section will hence give a brief overview on recent research related to tool use in robotics and is not exclusively limited to robot body schema as most research focuses either on learning the consistent correlation between different sensor modalities or sensorimotor control and only a minority directly addresses robot tool use.

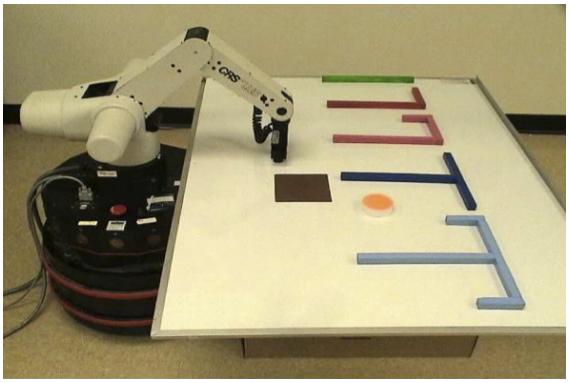
Nabeshima et al. (2006) presented a system realizing an adaptive body schema for tool use and the evaluation on a small tabletop humanoid robot. Whereas (Fitzpatrick and Metta, 2003) were able to detect alterations in the body, they provided no mechanism for adaptation. Nabeshima et al. emphasized the importance of tactile sensor and visual information under the aspect of timing ("temporal integration"). When using a tool, the robot has to recognize the tool as part of the body through the simultaneous observation of tactile and visual contact. In their system, the endpoint of the tool was detected by vision and through interaction with environment, that is, the feedback by hitting objects. They implemented their system with neural networks acting as associative memories on a simple robot arm with only two degrees of freedom. As it only used a stick as a tool, the new kinematics could be learned in form of a linear offset to the prior known kinematics. This differs from previously presented approaches, as the body structure of the robot, for instance, its kinematics, is not learned but assumed to be known in advance (that is, an *explicit model*). They refined the approach of tool assimilation in (Nabeshima et al., 2007) to detect collisions of the tool with the environment through the tactile sensors in the hand ('instantaneous sensor extension'). The inertia properties of the tool could be derived from observations while swinging the object. Further, they

generalized the functionality of the tool through visual and haptic interaction. The approach was evaluated in simulation where a planar robot with three degrees of freedom had to use several tools to retrieve an occluded object. The robot was able to learn shape and dynamic properties of these tools. In their most recent work, Nabeshima and Kuniyoshi (2010) presented a complete framework for marker-less visual-motor coordination and the identification of the dynamic properties of an attached tool. A robust method for the detection of an altered kinematics, by tool use for instance, were provided. The authors emphasize the importance of incremental and lightweight algorithms that enable online application on recent hardware. However, the identification of body parts still required a kinematic model of the robot but could be performed online and without further restrictions (i.e., no periodic movements) in contrast to prior work of Fitzpatrick and Metta (2003) or Natale (2005). This framework was successfully evaluated on a robot arm with six revolute degrees of freedom.

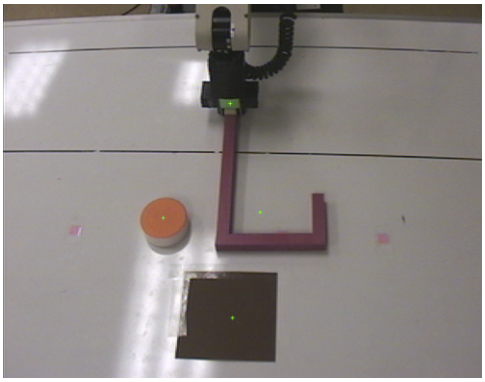
Kobayashi and Hosoe (2010) proposed a framework for a hierarchical and extendable body schema for tool use that did not require any prior knowledge on the kinematics and provided an efficient algorithm for planning in low-dimensional spaces. The detection of objects that can be used as a tool happened on a less abstract level and was based on the observation of synchronous motion between objects and the end effector—under the assumption that objects only move on contact with the robot. Once a contact between the robot and (possibly multiple) objects was established, this configuration was called a 'mode'. The framework determined the possible actions that maintain the current mode by means of a support vector machine (see Section 2.3.4) and parameterized the transition actions between different modes. The actions themselves were learned by instance-based statistical learning (see Section 2.3.4). Trajectory planning within and in between such modes, called 'plan shift', was a problem in lower-dimensional spaces that could be solved more easily. Although the work provided an intriguing view on goal directed planning and tool use, the framework was restricted to very simple scenarios, with planar objects and robots with circular end-effectors, and had only been evaluated in simulation. While the authors claimed the scalability of their method, the high dimensionality in a scenario with humanoid robots would require drastical modifications of the underlying representations.

The remaining recent efforts on tool use mostly focus on either learning and representing domain knowledge, that is, knowledge on why and which objects are suitable for what tasks. Brown and Sammut (2007) used a small mobile platform to learn the relation between tools and the goals they help to solve, and to generalize the properties that qualify an object as a tool. Their robots learned with an inductive logic programming (ILP) algorithm from demonstration of a similar agent. More recently in (Brown and Sammut, 2012), they proposed a system that also reduced the hypothesis space through active experimentation and used a novel action representation for symbolic planning, constraint solving and motion planning.

Stoytchev (2007, 2008) investigated tool affordance, that is the set of actions that tools can be used for due to their shape. Therefore, he learned tool use with an industrial robot arm with five degrees of freedom. The tools were of an abstract and simple shape and color-coded for easy visual detection and identification (see Figure 2.9). The robot had a programed repertoire of behaviors which it used to randomly explore the effect of an attached tool even if the tool breaks. The observations were organized in an 'affordance table' which related the tool to its usefulness to achieve a goal. After learning, the robot could select a tool by



(a) The robot can select a assortment of simple and color coded tools.



(b) The view the robot has on its wrist the tool and an attractor object.

Figure 2.9.: The experimental setup for learning tool affordances.
Source: (Stoytchev, 2008)

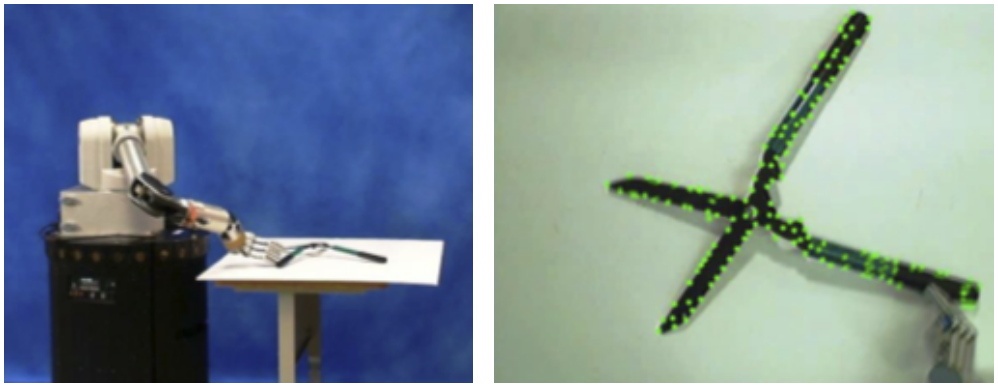


Figure 2.10.: Autonomous investigation of tools in an unstructured environment. *Source: (Katz et al., 2008)*

multiple nested line searches on the affordance table based on empirical heuristics. In their more recent work (Sinapov and Stoytchev, 2008), they concluded that the outcome of actions would be too high-dimensional for machine learning. Therefore, they organized the affordances in an ontology with an adaptive taxonomy to accurately describe the effects a robot can have on the environment and also detect these.

Although not directly related to tool use, research by Katz et al. (2008) should also be mentioned in this context. Their work included the autonomous investigation of articulated objects in unstructured environments by manipulation with a robot agent. By reinforcement learning and a symbolic representation, the robot acquired the skill to discover exploration strategies that lead to the efficient creation of a kinematic model of the explored object (see Figure 2.10).

Developmental robotics Metta et al. (1999, 2001) and Asada et al. (2001) suggested to apply the principles of the human ontogenesis also to artificial systems. Instead of the classical data collection, training and control cycle—often performed offline and manually—principles to create an autonomous and adaptive system were presented. One of the principles was called *"Modularity versus Integration"*. The traditional approach of modularizing a complex problem should be replaced according to the human development:

Newborn babies possess limited integrated movement capabilities in form of reflexes (Metta et al., 1999). These reflexes help infants to partially learn their body schema. Learning is hence balanced from simpler to more complex over time. The authors state that learning is, in general, an ill-posed problem and most learning could be subsumed as function mapping where the complexity is exponential in the number of dimensions (Bellman's curse of dimensionality (Bellman, 1960)). Tuning the parameters of a function approximation always means a trade-off between overfitting (good approximation – poor generalization) and over-smoothing (using a too simple model of the system) also called *Exploration–Exploitation dilemma*. Development is seen as the key to overcome this problem. The reduced sensori-motor abilities of an infant were consequently regarded as crucial to learning a body schema that becomes increasingly complex during its development. As a consequence, development was interpreted as an abstract control facility of the complexity of the learning process.

2.3. Sensorimotor learning in robotics

This thesis main focus lies on the development of new machine learning methods for sensorimotor learning and therefore the work related to this field will be presented in detail in this section. The main aspect of sensorimotor learning is the generation of models that optimally describe the complex relationships between sensor modalities and motor commands. In this context, two classes of models are predominant in the literature. The first class represents the *kinematics* of the own body and is consequently closely related to forward and inverse kinematics known in classical robotics. These models provide transformations between spaces that are crucial for planning in the external (Cartesian) space and execution in the robot's configuration space. While accurate kinematics knowledge is essential for action planning in Cartesian space, the second class of sensorimotor maps enhances greatly its execution, that is, the control of the robot. Such methods have to represent the dynamics of a robot. This knowledge is required for a maintained stability, energy optimized execution, and risk reduction through compliant control to name a few.

The idea of sensorimotor learning already exists for a long time starting with early approaches based on neural networks in last century's late eighties up to modern state-of-the-art machine learning techniques. This long history reflects in a vast literature available on this topic, and several recent surveys and comparative analyses (Nguyen-Tuong and Peters, 2011; Nguyen-Tuong et al., 2008; Sigaud et al., 2011). Notably, Sigaud et al. (2011) provide a comprehensive summary of sensorimotor learning methods and their application with a focus on online learning. Rather than attempting to duplicate this comprehensive overview, this section contains an excerpt that focuses on the analysis selected methods that lead to and justify the research of new techniques. As a consequence, those that are either the most prominent in the literature or those that are of importance in later chapters of this work are highlighted. This includes the methods that are used for evaluation purposes or those that reveal similarities with the proposed methods in this thesis. Therefore, *parametrized self-organizing maps*, *locally weighted projection regression*, *gaussian mixtures regression* and *gaussian process regression* will be studied in depth including their mathematical derivation while the remaining approaches will be addressed more in brief on few selected examples.

2.3.1. Fundamentals

For reasons of clarity and comprehensibility, the most important terms and concepts used to describe and classify the machine learning related to sensorimotor learning will be presented briefly.

Function regression: From a machine learning point of view, sensorimotor learning is almost always to be classified as an *inductive* and *supervised function regression*. Function regression creates a model of a *physical process* that can be described by a *latent function*. This manifests in an approximation which assigns a unique (continuos or discrete) output value to each element of a continuous input space. If a function with discrete outputs is searched for instead, this is referred to as a *classification problem*. In the context of function regression, *inductive learning* means that knowledge about the relation between the modalities is extracted from a set of examples according to an underlying assumption about the nature of the physical process—the so called *inductive bias*.

Supervised and unsupervised learning: The term *'supervised learning'* refers to the nature of the examples from which is learned. Here, an example consists of tuples containing the *input values (control signals)* and the *output values* observed from the underlying process (that is, the supervisor) in response.

In opposition, unsupervised learning aims at extracting structures or topologies present within a set of unlabeled data that is not separated into input and output variables (also called *stimuli*). It is also called *structural* or *topological learning* (Barreto et al., 2003) and closely related to *self-organization*.

Forward and inverse models: As mentioned, models learned with function regression map a control signal to a single and unique response sometimes together with a confidence indicator. They are called *forward models* (Massone, 1994) for their control/response nature. The *direct* or *forward kinematics* is an example of a forward model that can be expressed as a function

$$k(\boldsymbol{\theta}) = \boldsymbol{x}, \quad \boldsymbol{\theta} \in \Theta^{d_{\text{in}}}, \boldsymbol{x} \in \mathbb{R}^3 \text{ or } \boldsymbol{x} \in SE_3, \tag{2.1}$$

where $\boldsymbol{\theta}$ denotes a joint configuration and $\boldsymbol{x}$ an end-effector position (in the Cartesian space $\mathbb{R}^3$) or pose (out of the special Euclidean group SE_3). The natural number d_{in} denotes the number of the input dimensions, that is, the number of controllable joints or degrees of freedom (DoF).

Consequently, this excludes all redundant processes with a non-convex one-to-many relation (Bernstein, 1967), that is, the same input creates different outputs whose average is not a solution. Such processes are modeled by *inverse models*. The *inverse kinematics* of redundant robots is an important example for such a process that cannot be learned by function regression directly (see Jordan and Rumelhart, 1992)

$$k^{-1}(\boldsymbol{x}) = \boldsymbol{\theta} \tag{2.2}$$

If the number of input variables n is bigger than the output dimension this function is ill-posed and undefined. The same holds for the forward dynamics which, despite of the name, is ascribed to the class of inverse models much like the inverse kinematics in this context. Findings from brain research suggests that it is likely that the brain also encodes only forward models in its sensorimotor maps too

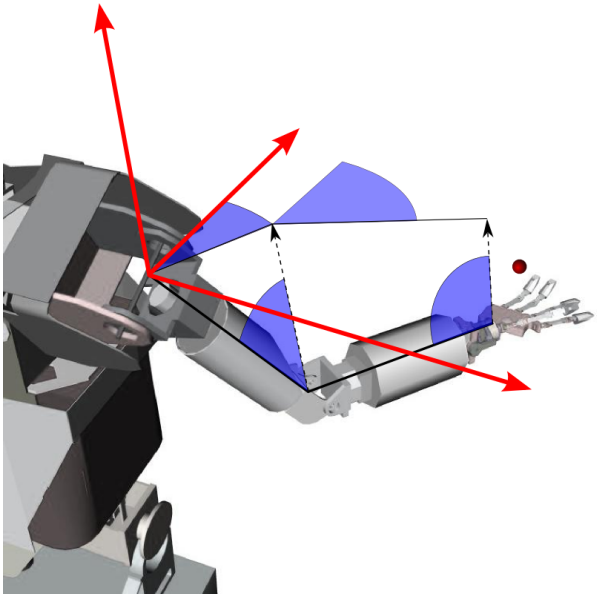


Figure 2.11.: Schematic illustrations of the ratio between joint angles in natural postures while pointing at a distant target on a humanoid robot as suggested by Soechting and Flanders (1989).

(see Wolpert et al., 1998).

Although some research addressed learning the inverse kinematics directly (Rolf et al., 2010b,a), learning an inverse model often involves a forward model. For instance, it is possible to learn the (invertible) first Taylor expansion (i.e., the *Jacobian matrix*) instead of the forward kinematics to achieve a local inverse model. This mainly becomes necessary, for instance, if the forward model is represented by a function that cannot be differentiated easily (Gaskett and Cheng, 2003).

Alternatively, a pure forward model can be inverted if the redundancy is resolved explicitly. This is a more universal approach as it puts one in a position to select how the redundancy is to be resolved. Psychological research suggests that humans resolve the redundancy in arm movements by means of cost functions (Cruse et al., 1990). In pointing gestures, a fixed ratio between the joint angles could be statistically observed in dependence of the location of the distant target (Soechting and Flanders, 1989; Asfour and Dillmann, 2003). Such a natural arm posture is exemplarily shown on the humanoid robot ARMAR-III in Figure 2.11. The inverse kinematics can also be numerically obtained by *resolved motion rate control (RMRC)* as presented by Whitney (1969) where *target velocity* is used to control the robot

$$J(\boldsymbol{\theta}) \cdot \Delta\boldsymbol{\theta} = \Delta\boldsymbol{x} \;\Leftrightarrow\; \Delta\boldsymbol{\theta} = \left(J(\boldsymbol{\theta})^T \cdot A \cdot J(\boldsymbol{\theta})\right)^{-1} \cdot J(\boldsymbol{\theta})^T \cdot \Delta\boldsymbol{x}, \tag{2.3}$$

where $J(\boldsymbol{\theta})$ is the Jacobian matrix that contains all partial derivatives of the current configuration with respect to all joints and A is a matrix that regulates the influence of the joint axis.

Bi-directional models: Note that there a third option of modeling sensorimotor maps as, strictly speaking, some of the algorithms presented later in this section belong into the field of unsupervised learning. With a trick, they can still be applied to sensorimotor maps when used to detect structures in supervised training data. Therefore, input and output values are combined into a single data vector

(also called *feature vector*).

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}^{in} \\ \boldsymbol{y}^{out} \end{pmatrix} \in \mathbb{R}^{d_{\text{in}}+d_{\text{out}}}, \text{ where } \boldsymbol{y}^{in} := \boldsymbol{\theta} \in \mathbb{R}^{d_{\text{in}}} \text{ and } \boldsymbol{y}^{out} := \boldsymbol{x} \in \mathbb{R}^{d_{\text{out}}} \tag{2.4}$$

where d_{out} denotes the output dimension (either $d_{\text{out}} = 3$ for positions or, for instance, $d_{\text{out}} = 6$ for the spatial pose). Learning in this form is sometimes called *'self-supervision'* (Barreto et al., 2003) and distinguishes itself from supervised learning because no error signals are explicitly calculated for learning. Consequently, it approximates the physical process as a bi-directional model rather than being limited to learning forward models only. Upon prediction, individual components of the feature can be designated as input or response values. This selection determines the 'direction' of the model. The problem of redundancy and multiple solutions still cannot be completely eluded and has to be dealt with when predicting in the 'inverse model' direction which involves the selection of an optimal solution with respect to some selection criteria.

Offline and online learning: Online learning refers to the ability of a learning system to learn and build a model during the regular operation of the robot. Offline learning refers to all opposing cases, for instance, if the robot has to perform special movements such that it cannot continue its regular application. Offline and online learning are closely related to the notions of *Batch* and *incremental learning*. If a system processes the complete training data set in one step this is called batch learning. In opposition, incremental learning successively processes the data to refine an already created model.

Generalization: The capability of making assumptions on unseen data is called *generalization* which can be divided into *interpolation* for prediction in between known data, or *extrapolation* for predictions that lie beyond the observation interval.

2.3.2. Taxonomy

The methods presented later in this section are mainly classified by two criteria: The distinction between *parametric* and *non-parametric models* on the one hand and, on the other hand, the distinction between *global* or *local* models.

Parameterization: A model is called *parametric* when its complete behavior can be described by a fixed and finite number of parameters. An artificial neural network, for instance, can completely be represented by the number of its neurons, the connection between them and the weights of the connections. Analogously, if there does not exit any fixed or finite number of parameters, the model is called *non-parametric*. To latter also belongs the group of *lazy learning* where a variable number of data points is not processed during learning and a prediction is directly made based on the examples. Learning that remember each training data are further called *memory-based* or *instance based learning*.

Type	Name	Scope	Parameterization	Comments
ANN	MLP	global	parametric	
	RBF	local	parametric	
	SOM	global	parametric	
	NGA, TPN	local	parametric	
	ITM	local	non-parametric	
	PSOM	global	parametric	
Statistical	SVR	global	parametric	lazy learning
	iRFRLS	local		
	LWR	local	non-parametric	lazy learning
	RFWR	local	non-parametric	
	LWPR	local	non-parametric	
	XSCF	local	non-parametric	
Stochastic	GMR	local	parametric	
		local	non-parametric	with modifications
	GPR		non-parametric	lazy learning
		local		stationary kernel
		global		non-stationary kernel

Table 2.1.: Taxonomy for classification of sensorimotor learning methods.

Scope: In *global learning*, one single model makes predictions for the whole configuration space—even if the required examples to cover the whole space accurately could not be collected. In contrast, *local learning* finds multiple simpler (usually linear) models that are valid only in a vicinity of the observed training examples. For a prediction, only the models that lie close to the query point contribute. Extrapolation is a property that naturally only to be found in global learning (Sigaud et al., 2011) Because of the linear models, local learning lacks the capability of extrapolation.

In the following sections, the most important methods from machine learning that were applied in the context of sensorimotor learning. In Table. 2.1, an overview over those method classified by the taxonomy is presented. Additionally, they are divided into *artificial neural networks*, *statistical* and *stochastic learning* and presented accordingly. Alongside, a selection of application in the field robotics will be presented in brief.

2.3.3. Artificial neural networks

Multilayer perceptrons and radial basis functions

Multilayer perceptions (MLP) are artificial neural networks in which the neurons are arranged in a directed graph. The neurons are organized in multiple layers where the first layer of d_{in} is directly connected to the inputs (the *input layer*) and the d_{out} neurons of the last layer (the *output layer*) provide the output of the network. In between those layer, there can be multiple hidden layers embedded. If the connections between layers is strictly proceding from the input layers through the hidden layers to the output layer, the network is called a *feed forward* network. The MLP is called a *recurrent network* if the connections between the neurons is less restrictive such that neurons can connect to neurons of the same or the preceding layer. A

neuron is modeled by its excitation level, that is, the weighted sum of its input signals and an activation function. For the i-th neuron in the j-th layer, the excitation can be expressed as the function

$$e_{i,j}(\boldsymbol{x}) = \sum_{k}^{n_{j-1}} w_k \cdot \theta_k = \boldsymbol{w}^T \cdot \boldsymbol{x}, \quad \boldsymbol{x}, \boldsymbol{w} \in \mathbb{R}^{n_{j-1}}$$

given an *stimulus vector* $\boldsymbol{x}$ and a *weight vector* $\boldsymbol{w}$ where n_{j-1} is the number of neurons in the preceding layer with $n_1 = d_{\text{in}}$. The activation function generates the response of the neuron with respect to its stimulation. Typically, strictly monotonically increasing functions such as the hyperbolic tangent

$$y_{i,j} = \tanh\left(e_{i,j}(\boldsymbol{x})\right)$$

or the sigmoid function

$$y_{i,j} = \left(1 + \exp\left(e_{i,j}(\boldsymbol{x})\right)\right)^{-1}.$$

Learning with such MLP is *global* and *parametric*.
However, the Gaussian bell can also be used as a localized activation function

$$y = a \cdot \exp\left(-\frac{e(x-b)^2}{2 \cdot c^2}\right), \quad a, b, c \in \mathbb{R}.$$

Networks using the latter activation function are called *radial basis functions (RBF)* networks, and have no hidden layers. Because of missing monotony learning with RBF still is *parametric* but the model is *local*.
At a first glance, neural networks like the multilayer perceptron seem to be the most obvious choice for the replication of a cerebral system, which is a neural circuit in reality after all. They also constituted the first attempts of sensorimotor learning, for instance in the work of Jordan and Rumelhart (1992) which also provides a good overview over the topic. This first impression, however, has to be revised as the learning of complex nonlinear functions with many inputs such as kinematics and dynamics, albeit possible (because they are proven universal function approximators (Hornik et al., 1989)), requires a very large number of neurons depending on the desired accuracy (Bishop, 2006). It is consequently very expensive in terms of robot movements and not the best choice for sensorimotor learning with many input dimensions.
Artificial neural networks are used, for instance, in the work of Nabeshima et al. (2006) as associative memories. In the experiments of Jordan and Rumelhart (1992), they were used to learn the forward kinematics and inverse dynamics of a planar robot with three degrees of freedom. Rolf et al. (2010c) used recurrent MLP networks to learn the full inverse kinematics of a humanoid robot, linking simultaneous stimuli together. Chinellato et al. (2011) used RBF networks to learn sensorimotor maps of inverse kinematics models that emerged from coordinated arm and eye movement. In their experiments, a humanoid robot learned these maps by exploring reaching and gazing actions.

Artificial neural networks: Self-organizing feature maps

Kohonen networks *Self organizing feature maps* or shorter *Self organizing Maps (SOM)* and *Kohonen networks*—named after Teuvo Kohonen who discovered/invented them in the early 1980ies (Kohonen, 1982)—embody a very different type of artificial neutral networks. They can also be used in a particularly

interesting approach to learning sensorimotor relations. Despite their limitation to discrete outputs and the fact that they are best suited for clustering problems in their original form, they perhaps bear the strongest relation of all techniques together presented in this thesis to the biological example. In the human brain, low-dimensional neural grid structures have been discovered that represent motor commands/body parts and map them into the peripersonal space.

The artificial neural networks described by Kohonen (1982) are *topology preserving*. This means that the SOM form a projection from the high-dimensional feature space (in this context, the combination of input and output space) to a discrete and lower dimensional space (a discrete multivariate interval or lattice) called *competitive* or *neural layer*. They have the remarkable property that they detect (i.e., learn) structures in the data and map them into a lower dimensional regular structure of connected neurons (King and Hwang, 1989; Caselli et al., 1994). Each neuron then represents a region in the feature space and neighborhood and distance relations between those regions are preserved. That way, relations can be detected and even the dimensionality can be reduced using a decidedly simple learning rule. As a typical (unsupervised) clustering algorithm, learning is unsupervised and sensorimotor learning consequently requires the combination of the input and output values into a single feature vector (see Eq. (2.4)).

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}^{in} \\ \boldsymbol{y}^{out} \end{pmatrix}, \quad \boldsymbol{y} \in \boldsymbol{R}^{d_{\text{in}}+d_{\text{out}}},$$

In other words, although it is an unsupervised learning algorithm it operates on data collected in a supervised data acquisition process. One benefit that results from that technique is that the strict separation between forward and inverse model is removed and the role of the features (i.e., components/dimensions of the data vectors) can be freely assigned for prediction (see Demers and Kreutz-Delgado, 1996). So if the joint angles are treated as outputs, and the Cartesian coordinates conversely as inputs, for instance, the SOM that models the inverse is

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{y}_{in} = \boldsymbol{x} \\ \boldsymbol{y}_{out} = \boldsymbol{\theta} \end{pmatrix}$$

Likewise, incomplete or missing data can be completed. For example, when the cartesian coordinates and all joints positions but the elbow are given, the elbow joint can be complemented by the model.

Self-supervision This mixture of the supervised and the unsupervised learning paradigms has been called *self-supervised learning* (Barreto et al., 2003). The model is constructed by first defining neurons rigidly connected to the nodes of a low-dimensional lattice that defines the neighborhood relations between the neurons. Each neuron is additionally connected to a point in the feature space called *weight* from now on:

$$\boldsymbol{w}_i = \begin{pmatrix} \boldsymbol{w}_i^{in} \\ \boldsymbol{w}_i^{out} \end{pmatrix}, \quad \boldsymbol{w}_i \in \boldsymbol{R}^{d_{\text{in}}+d_{\text{out}}}.$$

At the beginning of the learning process, these points are distributed randomly in the feature space. For each example or stimuli $\boldsymbol{y}$ observed during training, the one neuron is found, which weight lies closest—the

so-called *winning neuron*:

$$\boldsymbol{w}_{i^\star} = \arg\min_i \|\boldsymbol{y} - \boldsymbol{w}_i\|.$$

Its weight is adjusted such that the neuron becomes even more receptive to similar examples, that is, its weight is shifted towards the location stimulus. In addition, the neurons in the immediate neighborhood are also affected but to a lesser extent depending on their distance to the winning neuron in map space coordinates.

$$\Delta\boldsymbol{w}_i = \nu \cdot (\boldsymbol{y} - \boldsymbol{w}_i) \cdot h(i, i^\star),$$

where h is a distance metric and ν is a learn rate factor. The rule can can be further modified to repel points at a certain distance (by using the Mexican hat function) (Coiton et al., 1991). This rule is usually referred to as the "the winner takes it all"-rule and is inspired by the concept of *lateral inhibition* in the brain. The adaptation to the feature space is damped over the time by a factor that regulates to what extend weights are modified. Ultimately, this rule projects the topology embedded in the feature space onto the neutral structure. The problem of redundancy encountered with supervised function regression when learning forward models does not occur during learning—it is resolved by the combination of input and output which is always unique. However, predictions require the detection of the winning neuron and, for redundant functions, one has to choose one out of multiple possible candidates. This means that the problem of redundancy remains and has to be resolved just like in the case of supervised learning—at different time though.

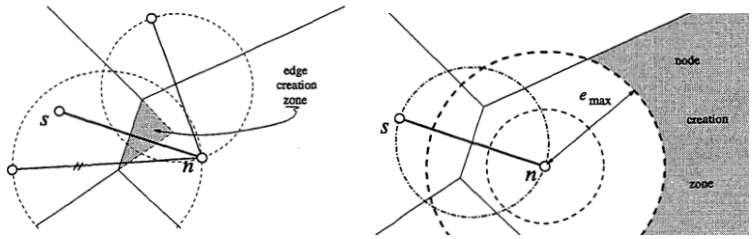
The next paragraph contains an overview on the achievements that could be made when using SOM and related algorithms for sensorimotor learning. Note that this has to be considered as an excerpt due to the large literature. A more comprehensive list and a very detailed introduction to self-organized learning can be found in (Barreto et al., 2003). The idea of learning the inverse kinematics with a SOM is quite old. For instance, it has been used in (Coiton et al., 1991) for learning the inverse kinematics of a robot arm with three non-redundant degrees of freedom. They applied a Mexican hat function for the determination of the lateral inhibition. Jones and Vernon (1994) showed that the same technique can be used to incorporate the camera calibration by directly relating the *retinal coordinates* of a stereo camera system (i.e., the pair of two-dimensional projections of the 3D cartesian coordinates) consequently evading the need for a reconstruction of the 3D coordinates of the end-effector. They also operated a robot arm with one prismatic and two rotary DoF and used infrared LEDs and filters to obtain the retinal coordinates. That way, they proposed a complete self-calibrating system.
This concept has been carried on in a more recent approach by Gaskett and Cheng (2003) that offered a complete integrated system. They used self-organizing maps with an underlying three-dimensional map space, to learn the kinematic models from uncalibrated vision of a humanoid robot with 30 DoF from which only 14 were controlled in this experiment. Their system is divided into two operation modes. It acts as an *open loop* controller when the hand is not visible relying on the already learned model. The *closed loop* control continues model learning when the hand is within the field of vision. In addition to sensorimotor learning, a motor-motor map was involved in the task to keep the end-effector always in the center of the view, that is, to keep the cameras fixated on the end-effector. This decomposition resulted in a reduction of

the input space which was also enfaced by mapping onto a three-dimensional neural structure embedded in the map space which is of lower dimensionality than the input space. The immediate benefit is that learning can be achieved in a comparably short amount of time. Further, they applied a relaxation measurement to achieve natural poses that also takes care that the object was only touched and not pushed away. Altogether, this demonstrated that the Kohonen rule is capable of capturing the most important characteristics of the topology in the feature space shown by an acceptable accuracy despite the dimensionality reduction.
Kihl et al. (1995) improved the learning process by sampling along randomly generated trajectories whereas usually random configurations were approached and observed. The work of Ritter et al. (1992) aimed at learning dynamic models for robots with three DoF.

Original Kohonen networks, however, expose several severe drawbacks—at least when applied to sensorimotor learning. One the one hand, the number of neurons is exponential in the dimension of the neuronal lattice. Both learning and prediction require the determination of the winning neuron, and they become hence very slow (i.e., computational time and quantity of training data) when mapping high-dimensional data. This problem is exponential complexity in the number of dimensions $\mathcal{O}(\exp(n))$ and is commonly called the "curse of dimensionality" (Bellman, 1960). On the other hand, the net produces strictly discrete output such that a high predictive accuracy also requires a very high density of neurons which amplifies the problems rising from high dimensionality. Several variations of the algorithms and extensions have been proposed over the yearrs to overcome these problems. The most important that are also related to sensorimotor mapping will be presented in the following. They can be separated into two classes: those without and those with predefined neighborhood relations.

Without predefined neighborhood relations Opposed to the original Kohonen networks, the idea of a fixed lattice which is responsible for concrete neighborhood relations between neurons is discarded and the connections are developed during the training phase. That way the prediction error can be reduced in regions of higher interest by dynamically increasing the density and connectivity in the related region without the immediate necessity to increase the overall granularity of the lattice. This improves the quality prediction and handles better the high dimensionality but contradicts the paradigm of a strictly global model.

The *neural gas algorithm (NGA)* presented by Martinetz and Schulten (1991) completely discards the lattice and the winning neuron. Only distances in the feature space influence the adjustments of the weights when learning from new observations (Behera et al., 1995) (i.e., the distance between weight and sample) leading to a very simple adaptation rule. The smaller this distance, the stronger is the adaptation to the example.
A more complex approach mark the *topology preserving networks (TPN)*. They are the result of the combination of the NGA with a competitive Hebbian learning rule (Hebb, 1949) that again projects into a low-dimensional space and hence preserves the topology. Weights are adjusted according to the rule inherited from the NGA (i.e., without consideration of the neurons location in map space) while the Hebbian rule is applied to the coordinates in the neural layer. The combination of topology preservation and increased accuracy bears advantages over the genuine SOM. An example for its deployment is a TPN that learns the model of a pneumatical driven robot using visual feedback presented in (Zeller et al., 1997)



(a) Edge insertion. (b) Neuron insertion.

Figure 2.12.: Rules for adding and removing new edges and neurons in *instantaneous topological maps* *Source: (Jockusch and Ritter, 1999)*

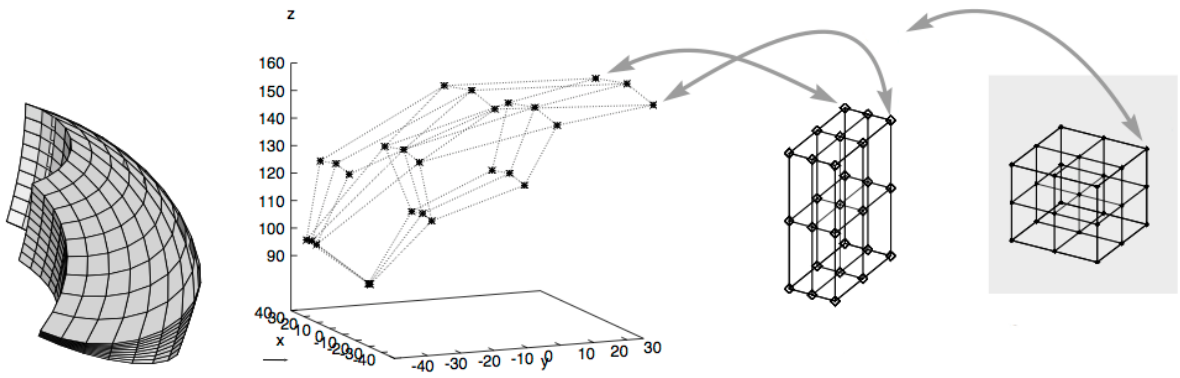
The *instantaneous topological maps (ITM)* by Jockusch and Ritter (1999) introduce the idea of undirected and unweighted edges between neurons to the Kohonen networks. That way, the algorithms overcomes degradation of the Kohonen network that arises if the stimuli are not statistically uncorrelated. This is then the case, for instance, if the feature space is not explored by approaching random configurations but along continuous movements resulting in temporarily connected stimuli from recorded trajectories. The ITM require adaptation of the edges connecting neighboring neurons during learning. Upon reception of a new stimulus, the two winning neurons are determined

$$n_1 = \arg\min_i \|\boldsymbol{y} - \boldsymbol{w}_i\|, \text{ and } n_2 = \arg\min_{j, j \neq n_1} \|\boldsymbol{y} - \boldsymbol{w}_j\|,$$

the weight w_{n_1} is adjusted towards the stimulus and an edge is created (if not already present) connecting the two winning neurons. Edges between the first winning neuron n_1 and its already existing neighbors are removed if the second winning neuron n_2 lies within the Thales circle over their line segment connecting their weights (see Figure 2.12a). If this leaves the neighbor neuron without any remaining edges, it is removed. A new neuron sf added if the stimulus $\boldsymbol{y}$ lies outside the Thales circle over the line segment spanned by $\boldsymbol{w}_{n_1}$ and $\boldsymbol{w}_{n_2}$ (see Figure 2.12b). The removal and creation of new neurons weakens the paradigm of global non-parametric learning. The ability to process temporarily connected stimuli renders the ITM well suited for learning from trajectories as done in (Vasquez et al., 2009).

Predefined neighborhood relations An alternative to abandoning predefined and fixed neighborhood relations to improve the prediction accuracy are variations of the Kohonen networks that add continuity to the neural net's output.

Local linear maps (LLM) defined by Ritter et al. (1989) add continuity to the output of the Kohonen network by combining neurons' weights with linear approximations of the manifold. In a vicinity of the weights, these linear models locally approximate the relation between the input $\boldsymbol{w}_i^{in}$ and output values $\boldsymbol{w}_i^{out}$ of the neuron's weight. Their construction rule can be found in (Ruiz de Angulo and Torras, 1997). This means that, despite the self-supervised Kohonen that is used for learning the topology, the linear models are again learned locally in a supervised process. After learning, this linear model is used to obtain a more precise



(a) The smooth manifold in the work space.

(b) The bi-directional mapping between workspace *(left)* and joint space *(middle)*. On the right, the connection to the neural lattice *(right)* is shown.

Figure 2.13.: Example of a kinematics model of robot finger with three degrees of freedom. *Source: (Walter, 1996)*

network output:

$$\boldsymbol{y}^{out} = \boldsymbol{w}_{i^\star}^{out} + A_{i^\star}\left(\boldsymbol{y}^{in} - \boldsymbol{w}_{i^\star}^{in}(t)\right)$$

The linear models approximate the Jacobian matrix at $\boldsymbol{w}_i$ vector so that the LLM results in a discrete first-order Taylor approximation of the sensorimotor map. That way, they show a great similarity with the locally weighted learning presented in the following section with the difference that the model is parametric and globa.l Notably, with this method, it is further possible to learn forward and inverse models alike. Its field of application, among others, includes calibration of a robot with three DoF in retinal coordinates obtained from stereo vision (Martinetz and Schulten, 1993), control of a robot with three degrees of freedom driven by pneumatic artificial muscles with taking the roles agonist and antagonist (Hesselroth et al., 1994), and the autonomous calibration of a space robot (Ruiz de Angulo and Torras, 1997).

The idea of self-organizing feature maps with continuous outputs has been pushed further by Ritter (Ritter, 1993). This effort led to the development of the *parameterized self-organizing maps (PSOM)* which can be used for efficient global and non-parametric sensorimotor learning. For that reason, they are be used for evaluation purposes in later chapters of this thesis (see Chapter 6.3). The main idea behind the PSOM algorithm is the interpolation of the neurons' weights in the feature space with a continuous manifold. The discrete neural lattice is now defined in a continuous space with coordinates $\boldsymbol{\lambda}$.
In contrast to the LLM, this approach does not require a modification of the learning rule nor a decomposition of the training data. Yet a higher precision can be achieved with a smaller number of neurons because of the curvature continuous manifold that interpolates the weights of the neurons. In the PSOM algorithm, the continuous manifold is realized by a multivariate Lagrange interpolation through the weights of all neurons (see Section 3.1.1 for polynomial approximation). Altogether, this means that the learning rule is not affected and the continuity is an interpretation of the network that becomes relevant when the network performs a prediction. On the downside, this choice of interpolation leads to polynomials of high degrees directly depending on the number of neurons per dimension. The results are higher computational costs, and unintended oscillations and asymptotic behavior (see Figure 3.1).
Still, the PSOM combine the benefits of the topological learning of the Kohonen networks with accurate con-

tinuous outputs. However, one of the problems mentioned earlier persists. Sensorimotor maps relate usually low-dimensional sensations (outputs) with high-dimensional motor-signals (inputs or proprioception). In case of the forward model of the kinematics of a humanoid robot, for instance, the perceived position of the hand in the frame of reference in the head depends on a large numbers of joint positions of the resulting kinematic chain. Consequently, the manifold to be learned cannot easily be mapped into the map space of a competitive layer unless its dimensionality equals the number of inputs (i.e., joints). Although this has been done by Gaskett and Cheng (2003) and Ruiz de Angulo and Torras (1997), it results in a dimensionality reduction and the lesser the dimensionality and number of inputs differ, the more detailed the input-output relation can be captured. As stated earlier, the number of neurons grows exponentially to the number of dimensions, hence capturing the complete relation is expensive in terms of computation time and quantity of the training data (i.e., robot movements). This problem is addressed by Walter (1996) for kinematic models, where the Kohonen learning rule is replaced by the direct assignment of the correct weights to the neurons by sampling the configuration space in a regular grid and the perceiving the respective outputs. That way, the topology is explicitly defined, and the underlying SOM degrades to an associative array (see Walter, 1998). While this appears to be a step backwards, it has to be considered that this also tremendously reduces the required quantity of training samples. It also comes at the price of a more complicated learning procedure and also reduces noise tolerance. In (Walter, 1996), the possibility of obtaining the optimal weights from randomly generated samples by linear optimization is suggested where the input values are used as map space coordinates effectively resulting in a supervised function regression. A rudimentary implementation of this idea is used in later chapters where a basic global non-parametric and incremental learning method is required (see Section 6.3.2). With their newest incarnation, named the *PSOM+* (Klanke and Ritter, 2005), a more elaborate variant has been introduced that also includes a smoothness metric to learn sensorimotor maps from noisy and not necessarily grid-organized training data. This efficiently prevents oscillation by the regulation of the curvature but also requires more hyper parameters that require task-specific tuning.

The PSOM variants gained popularity in sensorimotor learning. It has been applied for learning the inverse kinematics of robot fingers with three DoF (Walter and Ritter, 1995) (see Figure 2.13) or of a PUMA industrial robot arm with the same amount of DoF (Walter, 1996). Nolker and Ritter (2002) combined a LLM for learning visual features in 2D images with an inverse kinematics learned by a PSOM to recognize human hand postures. Cases with higher numbers of degrees of freedom were investigated by Ruiz de Angulo and Torras (2005a) where the kinematic model of PUMA industrial robot arm with six revolute joints was learned with slightly modified PSOM in a virtual decomposition (see Section 2.3.6). The PSOM+ are the most capable global non-parametric learning method suitable for high-dimensional sensorimotor learning. For that, the newly developed techniques presented in later chapters of this thesis are evaluated against them.

2.3.4. Statistical learning

Support vector regression

The *support vector regression (SVR)* is the equivalent of the *support vector machine (SVM)* in functions regression. In short, the SVM is a supervised classification algorithm that is based on the observation that many (if not all) classification problems that are not linearly separable can be transformed by a basis transformation $\varphi(\cdot)$ into a higher dimensional space in which they can be they can be linearly separated by a hyperplane. In addition, the hyperplane is obtained in a way that it optimally separates the labeled data. The margin that lies between the points classes is maximized with respect to its width during this optimization. Data points that lie on the boundaries of the margin are called support vectors as they determine the margin and they give the methods its name. Interestingly, it turned out not to be necessary to transform every data point explicitly into the high-dimensional space. In order to calculate the support vectors and the margin it is only necessary to calculate the scalar product in the transformation space which can be performed by a kernel function without explicit computation of $\varphi(\cdot)$

$$k(\boldsymbol{\theta}_i, \boldsymbol{\theta}_j) := < \varphi(\boldsymbol{\theta}_i), \varphi(\boldsymbol{\theta}_j) >,$$

for a latent function $f(\boldsymbol{\theta}) = \boldsymbol{x}$ (e.g., a kinematics). This implicit transformation is called the *kernel trick*, and through it, it even is possible to separate classes in infinite spaces. The Support Vector Regression transfers this concept to function regression and is a parametric model if the $\varphi(\cdot)$ maps into an finite d-dimensional feature space.

$$f(\boldsymbol{\theta}) = \boldsymbol{\varphi}(\boldsymbol{\theta})^T \cdot \boldsymbol{w} = \boldsymbol{x}, \quad \boldsymbol{w} \in \boldsymbol{R}^n,$$

where the model parameters in the weight vector $\boldsymbol{w}$ are a linear combination

$$\boldsymbol{w} = \sum_{i=1}^{n_f} \beta_i \varphi(\boldsymbol{\theta}_i)$$

of the transformed data points (i.e., support vectors) $\boldsymbol{\theta}_i$. Using a kernel this leads to

$$\hat{y} = \hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n_f} \beta_i < \varphi(\boldsymbol{x}_i), \varphi(\boldsymbol{x}) > = \sum \beta_i \cdot k(\boldsymbol{x}_i, \boldsymbol{x})$$

The approach is strongly related to the Gaussian process regression (GPR) that also applies the kernel trick (see Section 2.3.5). It has been applied to learn the inverse dynamics and has also been compared to locally weighted projection regression (LWPR) and GPR in (Fumagalli et al., 2010). Learning with SVR in general produces a global model of the latent functions. The method can be localized leading to the *incremental random features regularized least squares (iRFRLS)* presented by Gijsberts and Metta (2011). The authors evaluated the algorithm for learning the inverse dynamic model on simulated robot arms with four and seven degrees of freedom respectively. In (Droniou et al., 2012), the algorithm was compared to the extended classifying system for functions (XSCF) and LWPR (see the following paragraph) on the example of learning velocity kinematics. They showed that iRFRLS outperforms these methods.

Nonparametric local learning

In the previous sub-section, mainly parametric global models were presented. Despite of being the first methods for sensorimotor learning and being efficient for low dimensions, they were progressively replaced by methods better suited for dealing with the high dimensionality of the input space. The biggest problem is that a global model of a complex latent function that depends that many inputs requires a quantity of training data that is exponential in the number of inputs. The main idea behind local learning in general is the approximation of the sensor relation with a combination of linear models that are valid only within a vicinity of the observed training samples similar to the first taylor expansion of the target function. That way, these techniques are very similar to the LLM variant of the SOM with the only difference that not the complete domain of the target is approximated but only that portions that are actively explored. The immediate benefits that lead to the persistent success of local learning mainly are the observations that, on the one hand, learning starts immediately and on the other hand, that apart from being continuous, only very few assumptions have to be made about the target function and learning is very fast in general—the minimal number of training samples required to create the linear models equals the dimensionality of the input $\mathcal{O}(n)$. Compared to a PSOM+ with four neurons per input dimension, for instance, learning the forward kinematics of a robot arm with seven degrees of freedom requires only eight observations to build a single locally valid model and hence a minimally small portion of the robot's workspace—opposed to at least 4^7 for the complete space. Resulting from this locality rises the real strength of the local methods which lies in learning subspaces of the model and especially trajectories. According to (Sigaud et al., 2011), this is one main reason for the difficulty of comparing global and local models directly. And as a matter of fact, local methods also require huge amounts of training data when capturing the whole workspace—most likely more than a global learning—at least if a good inductive bias can be chosen for them. The boundaries between parametric and nonparametric local learning are less obvious. The individual linear models encountered in local learning are of course parametric. Their number is not limited though and is only regulated by the desired accuracy of the approximation or coverage of the action space. For that reason, they have to be considered nonparametric as opposed to the Gaussian mixture regression, for instance, which creates a parametric model that consists of a predetermined number of local models. In the following paragraph, the evolution of *locally weighted learning (LWL)* is presented from which the *locally weighted projection regression (LWPR)* finally emerged—the most prominent local nonparametric model in the literature. A summary of the algorithms within this class of algorithms is given in (Schaal et al., 2002). Despite its popularity, the extended classification system for function regression (XSCF), which developed independently, addresses the remaining issues the LWR are facing. However, they do have a lot in common and mainly differ in the way the receptive fields are constructed.

Locally weighted regression *Locally weighted learning (LWL)* encompasses several related algorithms. Among the initial attempts at local learning was the *Locally Weighted Regression (LWR)* (Schaal and Atkeson, 1994a). It is the most basic type of local learning and belongs to the class of *memory-based* and *lazy learning algorithms*. This means that no model (not even local ones) are created during learning.

Linear models are only implicitly created when the values of the latent function

$$\boldsymbol{x} = f(\boldsymbol{\theta}), \quad \boldsymbol{x} \in \mathbb{R}^{d_{\text{out}}}, \boldsymbol{\theta} \in \mathbb{R}^{d_{\text{in}}}$$

are evaluated. For a given query $\boldsymbol{\theta}_q$, the local linear model $q(\cdot)$ is defined by

$$q(\boldsymbol{\theta}_q) = \boldsymbol{\theta}_q^T \cdot \boldsymbol{\beta}.$$

Its model parameters $\boldsymbol{\beta}$ are obtained by

$$W \cdot B \cdot \boldsymbol{\beta} = W \cdot \boldsymbol{x} \Longleftrightarrow \boldsymbol{\beta} = (B^T \cdot W \cdot B)^{-1} \cdot B^T \cdot W \cdot \boldsymbol{x} \tag{2.5}$$

from the training data. They weighted by their distance to the query vector $\boldsymbol{\theta}_q$ encoded in the diagonal matrix W with

$$w_{i,j} = \begin{cases} \exp\left(-1/2(\boldsymbol{\theta}_i - \boldsymbol{\theta}_q) \cdot D \cdot (\boldsymbol{\theta}_i - \boldsymbol{\theta}_q)\right), & i = j \\ 0, & i \neq j \end{cases},$$

and the distances stored in

$$B = \begin{pmatrix} (\boldsymbol{\theta}_1 - \boldsymbol{\theta}_q)^T & 1 \\ \vdots & \vdots \\ (\boldsymbol{\theta}_{n_s} - \boldsymbol{\theta}_q)^T & 1 \end{pmatrix}.$$

The prediction results from the insertion of x in all possible models hence the quadratic complexity. This procedure is influenced by a single hyper parameter h embedded into the distance metric D,

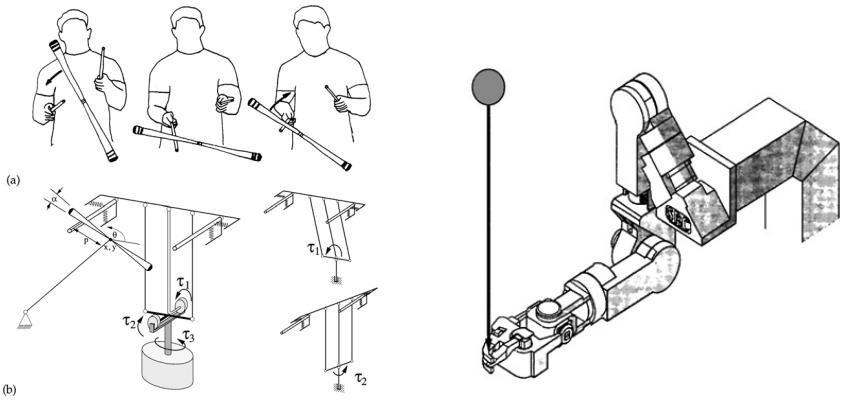
$$D = h \cdot \begin{pmatrix} 1/\sigma_1^2 & & 0 \\ & \ddots & \\ 0 & & 1/\sigma_m^2 \end{pmatrix},$$

where the σ_i^2 denote the variance of input valued in the i-th component of the input data $\boldsymbol{\theta}_i$. Its value is selected from a set of possible values using "leave-one-out" cross validation. This means that in a data set $\mathscr{T}$ of n training samples is divided into n subsets $\mathscr{T}_i$ each reduced by a single different data point:

$$\mathscr{T}_i := \mathscr{T} \setminus \{\boldsymbol{x}_i\}$$

For each set $\mathscr{T}_i$, a LWR predicts the value one data point $\boldsymbol{x}_i$ that has been left out and the mean squared error (MSE) over all n sets $\mathscr{T}_i$ is computed. The procedure is repeated for all candidate values for the hyperparameter h and the optimal value is chosen with respect to the smallest MSE. In the context of sensorimotor learning, LWR has been successfully applied by Schaal and Atkeson (1994b) who applied it for the control of a robot with two degrees of freedom playing devil sticks (see Figure 2.14a).

The concept introduced by LWR was then refined in the *Locally weighted partial least squares (LW-PLS)* (Schaal et al., 1998) using *partial least squares regression (PLS-1)* (Wold, 1975; Frank and Friedman,



(a) Robot playing devil stick. (b) Robot balancing a pole stick.

Figure 2.14.: Dexterous manipulation with locally weighted learning. *Source: (Schaal et al., 2002)*

1993) which enables learning data with high-dimensional input spaces, above all, with irrelevant and correlated input dimensions. Due to the otherwise necessary matrix inversion (see Eq. (2.5)) this approach is also responsible for a significant increase in numeric stability. Partial least squares is also an integral part of the LWPR algorithm introduced later in this section and will be explained more in detail in that context.

So far, locally weighted learning consisted only of memory based and lazy learning methods. This bears the disadvantage that at any time all encountered training samples have to be remembered (i.e., kept in memory)—a number that is bound to grow very quickly for redundant robots with many input dimensions. Because of this observation, the *receptive field weighted regression (RFWR)* was developed by Schaal and Atkeson (1997) as an incremental variant of LWL. It uses *recursive least squares (RLS)* (Ljung and Söderström, 1985) to incrementally create local models. These can be updated when new data is observed without the need of keeping all samples constantly in memory. In addition, for each model, all data points during learning and prediction are weighted by the influence of a receptive field, that is, a Gaussian kernel with its center $\boldsymbol{\gamma}_i$ in the input space and a designated covariance matrix D_i (Gaussian kernel).

$$w_i = \exp\left(-1/2(\boldsymbol{\theta} - \boldsymbol{\gamma}_i) \cdot D_i \cdot (\boldsymbol{\theta} - \boldsymbol{\gamma}_i)\right)$$

A prediction is obtained by the weighted sum

$$\boldsymbol{x}_q = \frac{\sum_{i=1}^{n_m} w_i \cdot \boldsymbol{\theta}_q^T \cdot \boldsymbol{\beta}_i}{\sum_i^{n_m} w_i}$$

The receptive fields are stationary, that is, their centers $\boldsymbol{\gamma}_i$ are not modified once created. However, their shape D_i, vary with each newly encountered observation according to a rule that it shares with the LWPR. The algorithm starts with an empty list of receptive fields. Every time a new observation is processed, the activation w_i of every receptive field is computed (Eq. (2.3.4)). This value is required to update the shape D_i of already existing receptive fields. A new receptive field is only then created if none of the activation exceeds given threshold. In this case, which occurs already with the very first example, the current input $\boldsymbol{\theta}$ is used as the center in the input space and a default covariance is chosen. The algorithm is shown in pseudo code in Algorithm. 1. The complete procedure can be found in (Schaal et al., 2002). With RWRF, it was possible to learn the control of a industrial robot arm for a dexterous task. The robot balanced a pole stick

Algorithm 1: The update rule for the receptive fields in *receptive fields weighted regression* and *locally weighted projection regression* (Schaal et al., 2002).

Initialization: Number of receptive fields $n_m \leftarrow 0$;
for *every new training sample* $(\boldsymbol{x}, y)$ **do**
  for $i = 1$ *to* n_m **do**
    Activation $w_i \leftarrow \exp\left(-1/2(\boldsymbol{x}-\boldsymbol{c}_i)\cdot D_i \cdot (\boldsymbol{x}-\boldsymbol{c}_i)\right)$;
    Update shape D_i;
  if $w_i < w_g,\ \forall i \in \{1,\ldots,n_m\}$ **then**
    `// Add new receptive field:`
    $n_m \leftarrow n_m + 1$;
    $c_{n_m} = \boldsymbol{x}$;
    $D_{n_m} = D_0$;

upright using all of its seven degrees of freedom (Schaal et al., 2000) (see Figure 2.14b). The work of Nori

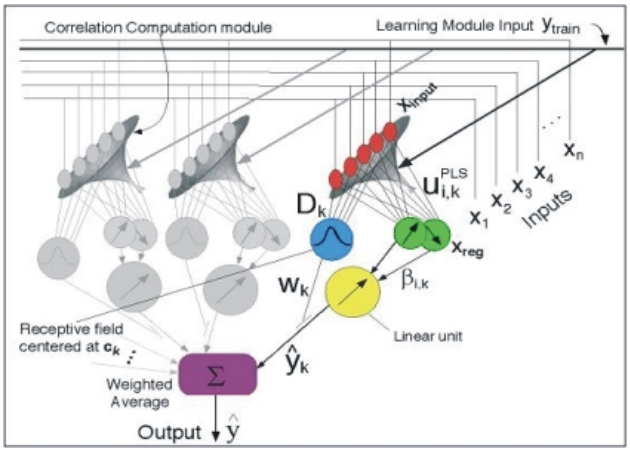


Figure 2.15.: The schematic description of the locally weighted projection regression (*Source:* Vijayakumar and Schaal, 2000).

and Natale et al. in (Nori et al., 2007; Natale et al., 2007) differed from classical approaches by offering a method for neglecting the calibration of the camera and the kinematic parameters. The key is that the robot has to fixate the object that is to be reached for hence implicitly encoding the world position of the objects within the neck and eye degrees of freedom. Then joint values of the head have to be mapped to joint values of the arm — a motor-motor map has to be learned. The method does not cope redundant robot though, the authors refer to (Lopes and Santos-Victor, 2006). Open and closed loop approaches are both included in this work as the closed loop is used as soon the visual feedback becomes available. A neural network (receptive field weighted regression mode) is used to calculate a rough prediction for the open loop control. Redundancy is resolved by containing joints. Learning takes 200 points for a 4 degrees of freedom arms. Closed loop learning maps the eye-to-hand Jacobian again by neural nets.

The probably most popular and most widely applied algorithm for learning sensorimotor maps with non-parametric local learning, however, is the *locally weighted projection regression (LWPR)* by Vijayakumar and Schaal (2000). It combines the incremental nature of its predecessor RWRF with the partial least squares regression of LWPLS which has been enhanced to allow incremental updates of the linear models. LWPR

has been applied to numerous use-cases in robot control, for instance, in (Lammert et al., 2013; Sun de la Cruz et al., 2011; Stalph et al., 2010; Salaün et al., 2009; Sun de la Cruz et al., 2010; Hoffmann et al., 2009; D'Souza et al., 2001) to name just a few, and has often been compared to competitive methods (Droniou et al., 2012; Stalph et al., 2010; Sun de la Cruz et al., 2010; Nguyen-Tuong et al., 2008; Flentge, 2006). A very well documented and freely available implementation with interfaces to various programming languages probably holds a big share of responsibility for its wide popularity. Some aspects of this algorithm partly inspired the development of the new methods presented in later chapters—above all, the incremental partial least squares regression and its ability to deal with high input dimensions. For that reason, it will be described in a more detailed and mathematical way in the following paragraph.
As we will see later, the property of LWPR that it distinguishes it most from the afterwards introduced XSCF is the procedure used to create new receptive fields and how they finally are distributed in the input space. As mentioned in the paragraph of the RFWR, the LWPR algorithm decides whether it is already sufficiently influenced by a receptive field or whether to create a new one for every new data point. That is, if

$$w_i < w_g,\ \forall i \in \{1,\ldots,n_m\} \tag{2.6}$$

holds, the point is sufficiently covered by the model/field M_i. However, the shape of the receptive field might have to be adjusted to increase its influence in this region according to Algorithm. 1. Consequently the same as before for the RFWR applies: If a newly presented example cannot be explained by any of the existing models (i.e., activates no receptive field), a new receptive field is created with its center at exactly this point and an initial default covariance. Unlike the covariance, which is updated in a gradient descent, the center of the receptive field cannot be altered once created (although models and receptive fields can degrade by a forgetting factor over time). The mechanism that regulates the distribution and creates them on demand is very straightforward and easy to implement. This comes at the price of flexibility as it cannot consider the whole data set and choose the optimal distribution though. Outliers, for instance, may result in unnecessary receptive fields that interfere with correct data. Another attribute of LWPR that is often criticized is the large number of hyper parameters such and the sensitivity that make parameter tuning challenging (Sigaud et al., 2011).

Extended classifying system for function regression (XSCF)

A more robust and recent approach tries to evade the problems accompanying LWPR: The *extended classification system for function regression (XSCF)*—although closely related—does not originate from the development line of LWL. It is an advancement of the extended classifying system (XSC) that allows function regression with local linear models in a very similar way as the locally weighted learning methods do. Consequently, we will focus on the differences to LWPR in the following. The first eminent difference to LWPR lies in the receptive fields. Models do not contribute to the weighted sum at all when the input lies outside the region of influence—in LWPR, every model contributes to the sum albeit often only very little. Another difference is that the choice receptive fields is not limited to ellipsoid shapes (the equivalent to the Gaussian kernel) even if they are prevailing in sensorimotor-related learning and allow the best comparison. Local models are learned with recursive least squares (RLS) optimization that also is incremental and

assumably faster than the partial least squares method. It cannot not equally profit from a dimensionality reduction whereas the authors argue that this is mainly important when learning trajectories. The biggest distinction, however, is the distribution of the receptive fields. In LWPR their creation follows a relatively simple rules—they are created in the data points if those cannot sufficiently be explained by existing models. XSCF uses a different approach. A steady-state genetic approach determines the allocation of receptive fields that optimally explain the training data. In an artificial evolution, receptive fields are created as children of existing fields and selected such that in the end the data is optimally explained by as few local models as possible. These theoretical differences have also been evaluated for applications in the robotics domain. In (Butz et al., 2009; Butz and Herbort, 2008; Stalph et al., 2009) the XCSF was used to learn a simulated robot arm with four degrees of freedom. Later (Stalph et al., 2010), they compared their approach to the LWPR.

In total, XSCF and LWPR (and the other locally weighted learning approaches) have many concepts and properties in common and it has been suggested to combine their strength into a single practice. Nonparametric and local learning methods, in general, expose their real strength when the input dimensionality of the sensorimotor map is large and not the whole input space is to be covered but only a small portion of it or low-dimensional subspace. This makes them the ideal choice for learning trajectories, for instance, in imitation learning.

2.3.5. Probabilistic learning

Gaussian mixture regression (GMR)

In machine learning, a *Gaussian Mixture Model (GMM)* is used for probabilistic learning and refers to an inductive bias with the assumption that the observed data is distributed according to a mixture of Gaussian distributions (i.e., models). This means that in a data set, each observation belongs to one of a finite number of Gaussian models (normal distribution) similar to the receptive fields in locally weighted learning. Formally, let

$$Y = \{Y_1, \ldots, Y_n\}, \quad Y_i \in \boldsymbol{R}^d,$$

be a set of n random variables that are distributed by a mixture of m Gaussian models. Then there also exist n *latent* (i.e., *hidden*, unobservable) random variables

$$Z = \{Z_1, \ldots, Z_n\}, \quad Z_i \in \{1, \ldots, m\},$$

indicating the association of an obervation to one of the Gaussian models such that for n given observations $\boldsymbol{y} = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_n\}$ and latent variables $\boldsymbol{z} = \{z_1, \ldots, z_n\}$, the probability density for the observations is

$$p(\boldsymbol{y}) = \bigcap_i P(Y_i = \boldsymbol{y}_i) = \prod_i^n p(\boldsymbol{y}_i),$$

where the density for each observation is

$$p(\boldsymbol{y}_i) = \bigcup_j P(Y_i = \boldsymbol{y}_i \wedge Z_i = j) = \sum_j^m P(Z_i = j) \cdot p(\boldsymbol{y}_i | Z_i = j),$$

where the density for each Gaussian model

$$p(\boldsymbol{y}_i | Z_i = j) = \mathcal{N}(\boldsymbol{y}_i; \boldsymbol{\mu}_j, \Sigma_j)$$

is a multivariate normal distribution with mean $\boldsymbol{\mu}_j$ and covariance matrix Σ_j. In this definition all $p(\boldsymbol{y}_\square) \mapsto P(Y_\square = \boldsymbol{y}_\square)$ are (continuous) probability density functions and $p(z_\square) \mapsto P(Z_\square = z_\square)$ (discrete) probability mass functions.

The parameters that completely define a Gaussian mixture model are the parameters of the individual Gaussian Models and the prior probability distribution of the Gaussian models $p(z)$ forming a parameter set $\mathcal{M}$

$$\mathcal{M} := \{p(z), \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_m, \Sigma_1, \ldots, \Sigma_m\}.$$

Gaussian mixture models, in general, are used for *clustering* and *classification*. This means that, after having learned the model, the conditioned probability distribution $p(z|Y = \boldsymbol{y}^\star)$ given an observed variable $\boldsymbol{y}^\star$ is computed, and the most likely Gaussian model $z^\star$ is obtained.
Further, it is an *unsupervised learning* process as the data is not divided into input and output data. Less frequently, however, it is also used for general function regression which is then called *Gaussian Mixture Regression (GMR)*. Here, the coordinates of the data points are divided into d_{in} input values and d_{out} output values *after* the learning process (i.e., *self-supervised learning*, Section 2.3.3). In the case of learning a FK, for instance, the data are joint angles and the end-effector positions

$$\boldsymbol{y} = (\boldsymbol{\theta}, \boldsymbol{x})^T, \quad \boldsymbol{\theta} \in \boldsymbol{R}^{d_{\text{in}}}, \boldsymbol{x} \in \boldsymbol{R}^{d_{\text{out}}}.$$

Learning a GMM is obviously a case of *parameterized* (or *model-based*) *learning* as observed data is used for deriving the model parameters and neither for inference nor classification. In the next sections, the most prominent learning algorithm for GMM—*Expectation Maximization*—and the Gaussian Mixture Regression are presented.

Expectation Maximization The standard algorithm used for learning mixtures of Gaussians is called *Expectation Maximization (EM)*. This algorithm finds the parameter set $\hat{\mathcal{M}}$ for the Gaussian models that maximizes the expectation $\mathbb{E}$ of the log-likelihood:

$$\hat{\mathcal{M}} = \arg\max_{\mathcal{M}} (\mathbb{E}(\log l(\mathcal{M}|\boldsymbol{x}, \boldsymbol{z}))). \tag{2.7}$$

As it is not possible to express the optimal parameters in a closed form, the EM-algorithm iteratively searches for a solution. In brief, the algorithm performs two steps in each iteration starting with an ar-

bitrarily chosen initialization of the parameters. The *expectation step* estimates for each given data point and each Gaussian model the probability that the data belongs to this model. In the *maximization step*, new model parameters are calculated that match best this estimated distribution. The algorithm terminates as soon as the improvements after an iteration drop below a certain threshold. Although the algorithm is guaranteed to find better parameters in every step, it is prone to getting caught in local minima, giving the initial assignment of the parameters a special importance.

Gaussian Mixture Regression (GMR) Learning the parameters of Gaussian Mixture Models imposes a clustering on the observed data points. This section shows how this learned clustering can also be used to predict smooth and continuous functions. Therefore, the coordinates the data points are divided into input and output dimensions. Given a data random variable $Y \in \mathbb{R}^d$ in kinematics, for instance, the coordinates can be separated into a new random variable for input dimensions $T \in \mathbb{R}^{d_{\text{in}}}$ (for joint values) and one for the output dimensions $X \in \mathbb{R}^{d_{\text{out}}}$ (for the end-effector position)

$$X = (T,X)^T, X \in \mathbb{R}^{d_{\text{in}}+d_{\text{out}}}$$

For prediction, a data point $Y = \boldsymbol{y}^\star$ with

$$\boldsymbol{y}^\star = (\boldsymbol{\theta}^\star, \boldsymbol{x}^\star)^T,$$

is assumed where only the joint values $\boldsymbol{\theta}^\star$ are known. The forward kinematics for $\boldsymbol{\theta}^\star$ can then be predicted by first calculating the probability distribution of $\boldsymbol{x}^\star$ given the learned model parameters $\mathscr{M}$,

$$p_{\mathscr{M}}(\boldsymbol{x}^\star|\boldsymbol{\theta}^\star) = \sum_j^m P_{\mathscr{M}}(Z^\star = j) \cdot p_{\mathscr{M}}(\boldsymbol{x}^\star|\boldsymbol{\theta}^\star, Z^\star = j), \tag{2.8}$$

where $Z^\star$ again refers to the non-observable Gaussian model that $\boldsymbol{x}^\star$ belongs to. The conditional probability distribution assuming a a Gaussian model $Z = j$ essentially is a d-variate normal distribution that is intersected with a subspace by a projection

$$\pi_{\boldsymbol{\theta}^\star}\Big((\boldsymbol{\theta},\boldsymbol{x})\Big) = (\boldsymbol{\theta}^\star, \boldsymbol{x})$$

for a constant $\boldsymbol{\theta}^\star$ The result of this intersection is again a normal distribution $\mathscr{N}^X$ which is d_{out}-dimensional

$$p_{\mathscr{M}}(\boldsymbol{y}^\star|\boldsymbol{\theta}^\star, Z^\star = j) = \mathscr{N}_j^X(\boldsymbol{\mu}_j^X, \Sigma_j^X).$$

and is called a emphconditioned distribution. The parameters of $\mathscr{N}^X$ can be obtained easily from the parameter set $\mathscr{M}$. First, we have to decompose the components of the known parameters $\boldsymbol{\mu}_j$ and Σ_j

$$\boldsymbol{\mu}_j = ({}_\theta\boldsymbol{\mu}_j, {}_x\boldsymbol{\mu}_j)^T, \qquad \Sigma_j = \begin{bmatrix} {}_\theta\Sigma_j & \boldsymbol{\sigma}_j^T \\ \boldsymbol{\sigma}_j & {}_x\Sigma_j \end{bmatrix}, \tag{2.9}$$

Then, the intersection $\mathcal{N}_j^X(\boldsymbol{\mu}_j^X, \Sigma_j^X)$ then is determined by

$$\begin{aligned} \boldsymbol{\mu}_j^X &= {}_\theta\boldsymbol{\mu}_j + \boldsymbol{\sigma}_j^T \cdot {}_\theta\Sigma_j^{-1} \cdot (\boldsymbol{\theta} - {}_\theta\boldsymbol{\mu}_j), \\ \Sigma_j^X &= {}_x\Sigma_j - \boldsymbol{\sigma}_j \cdot {}_\theta\Sigma_j \cdot \boldsymbol{\sigma}_j^T. \end{aligned} \tag{2.10}$$

The prediction $\boldsymbol{x}_j^\star$ of the Gaussian model given the joint values $\boldsymbol{\theta}^\star$ now is the most likely vector, that is, the mean $\boldsymbol{\mu}_j^X$ of named distribution. Finally, the sum in Eq. (2.8) is solved for $x^\star$. An important property of the GMM and other probabilistic approaches is that the covariance matrix $\Sigma^\star$ can also be computed easily. It is a very good indicator for the confidence of the prediction.

Figure 2.16 shows the results of an evaluation of the GMM where the x-coordinate of a robot with two orthogonal axis has been learned from a training set

$$\mathcal{T} = \{(\theta_{1,i}, \theta_{2,i}, x_i)^T : i = 1, \ldots, 8000\}$$

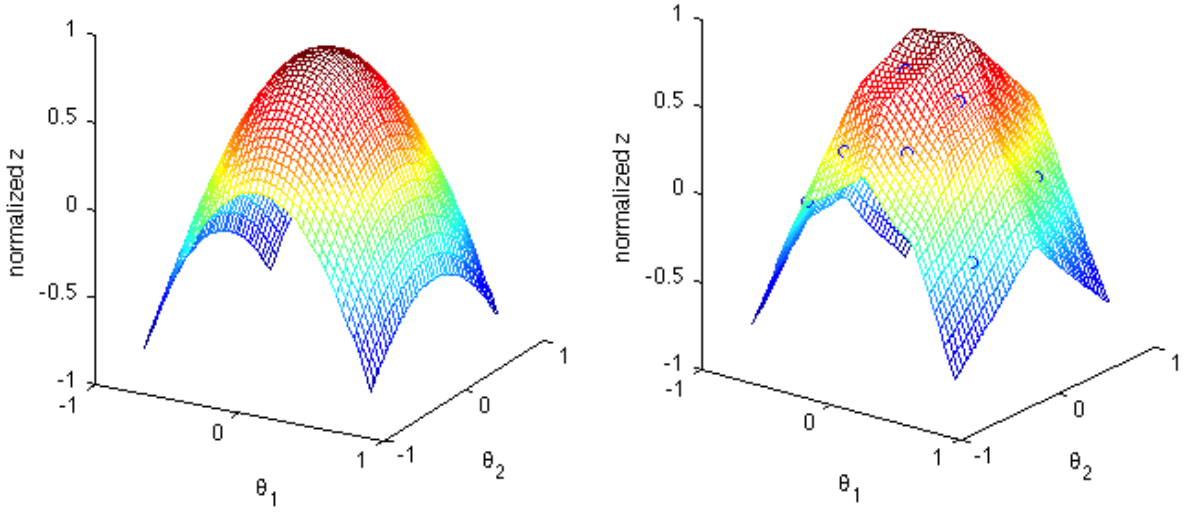
an GMM model of ten models. This number is the only hyper parameter in the GMR approach.

In general, Gaussian mixture regression is parametric learning as the number of Gaussians models is constant. In (Lopes and Damas, 2007; Damas and Lopes, 2007) however, Damas and Lopes presented a modification of the of the expectation maximization process such it can be learned incrementally and online. Further, the number of Gaussian models is not fixed anymore. New models are added to the mixture whenever a training sample is observed that cannot be explained if the probability for belong to one of the models is smaller than a threshold. That way the algorithm resembles very much to the locally weighted progression regression with the exception that the centers of the Gaussians are updated by the EM algorithm. Recently, Chatzis et al. (2012) presented another non-parametric variant of GMR-based Dirichlet process mixture models.
Similar to non-parametric statistical learning, Gaussian mixture models are best suited for learning along trajectories. Calinon et al. (2007), for instance, applied the GMR for programming-by-demonstration (similar to (Chatzis et al., 2012)). However, it could also be applied to directly learn the inverse kinematics of a planar robot with three degrees of freedom as shown in an early work of Ghahramani (1993). It has also been use to learn the whole configuration space (see Figure 2.16) and in combination with a kinematic decomposition (Ulbrich et al., 2012b)).

Gaussian process regression

Definition The easiest approach to understanding Gaussian processes is to think of them as a generalization of multivariate Gaussian distributions that can hold an infinite number of random variables. So they are formally defined as the distribution of set of random variables X over an arbitrary index set $\mathcal{S}$

$$X = \left\{ X_s : s \in \mathcal{S} \right\} \sim GP_{\mathcal{S}}$$



(a) The kinematic function.

(b) The predicted function with the centers of the Gaussian models.

Figure 2.16.: Example of a forward model learned with GMR. It shows the x-coordinate of the end-effector in dependance of the joint angles of a robot with two degrees of freedom. The composition of linear models is observable.

which must fulfill the single requirement that each finite subset of the random variables is distributed by multivariate Gaussian.

$$\forall \mathcal{S}_n = \left\{s_1, \ldots, s_n\right\},\ n \in \mathbb{N},\ \mathcal{S}_n \subseteq \mathcal{S}: \quad \exists\, \boldsymbol{\mu}, \Sigma : (X_{s_1}, \ldots, X_{s_n}) \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma)$$

The most simple example of a Gaussian process is, consequently, the multivariate Gaussian distribution itself with—it is defined over the finite index set $\mathcal{S} = \{1, \ldots, n\}$ where n is the number of variables. Each subset of its random variables is also normally distributed by definition (the so-called *affine property* of multivariate Gaussians).

However, the index set does not necessarily have to be finite or even countable and can be multi dimension. It could consequently be, for instance, the real numbers $\mathbb{R}$ or a vector space $\mathbb{R}^n$. When there is an infinite number of random variables, it is impossible to describe the distribution by a single mean $\boldsymbol{\mu}$ and or covariance Σ. The continuity in the indices of the random variables requires the mean and covariance to be defined as *continuous functions*—the *mean function* $\mu(s)$ and *covariance function* $k(s,t)$ where $s,t \in \mathcal{S}$ are the indices of two random variables out of X. These two functions are sufficient to completely describe a Gaussian process.

If used in the context of regression in machine learning, nearly exclusively finite numbers of random variables are encountered—each variable represents training sample from a physical process. That way, one can use the calculus known for multivariate Gaussians. For instance, predictions are made by conditional distributions similar as in the GMR (see Eq. (2.10)). In the example of kinematics this means that the training set

$$\mathcal{T} = \left\{(\boldsymbol{\theta}_i, \boldsymbol{x}_i) : i =, 1 \ldots, m\right\}$$

defines an index over all observed joint angles

$$\mathscr{S}_\theta = \left\{ \boldsymbol{\theta} : (\boldsymbol{\theta}, \boldsymbol{x}) \in \mathscr{T} \right\}$$

and the end-effector positions are the random variables

$$X = \left\{ X_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \mathscr{S}_\theta \right\} \sim GP_{\mathscr{S}_\theta}$$

distributed by the Gaussian process. Together with a mean and covariance function $m(\cdot)$ and $k(\cdot,\cdot)$, the Gaussian process can then be interpreted as the random distribution of functions over index set $\mathscr{S}_\theta$

$$f \sim GP(m,k)$$

The greatest strength of the Gaussian process regression lies in the freedom of the definition the covariance function which is only restricted to be positive semi-definite. This allows the inclusion of different degrees of a-priori knowledge about the nature of the underlying physical processes that one is trying to model. It also makes the Gaussian Processes Regression a very powerful tool that has been becoming increasingly popular in machine learning for robotics recently and many prominent instance-based machine learning methods can be expressed in terms of a Gaussian process as we will see later in this chapter. On the other hand, the weakness of the GPR is also related to the covariance function. It is instance-based learning and, as such, requires the calculation of the covariance of all known random variables, that is, all observed samples. Furthermore, the resulting covariance matrix has to be inverted for making predictions which cannot be done with a complexity less than $\mathscr{O}(n^3)$.

Kernels and mean functions The covariance function is also often referred to as *kernel*. Gaussian process regression can be used for *Bayesian linear regression* when using a *linear kernel*

$$k_l(s,t) := \langle \varphi(s), \varphi(t) \rangle^n, \quad s,t \in \mathscr{S},\ n \in \mathbb{N}_+. \tag{2.11}$$

with $n = 1$ where a basis transformation $\varphi(\cdot)$ can be applied to the indices. For $n > 2$ the kernel is called *polynomial*.
The most popular kernel in the context of learning robot models, is the *squared exponential function*

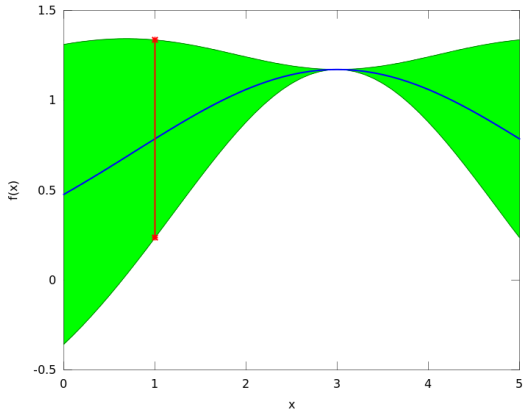
$$k_{se}(x,y) = e^{-a\cdot(x-y)^2}, \quad a \in \mathbb{R}^+, \quad x,y \in S, \tag{2.12}$$

where a is weighting hyper parameter. A Gaussian process with zero mean and the squared exponential kernel is a distribution of smooth and differentiable functions which resembles greatly renders it especially well suited for general function approximation. Alternatively, it can be defined in dependence of the distance between two random variables:

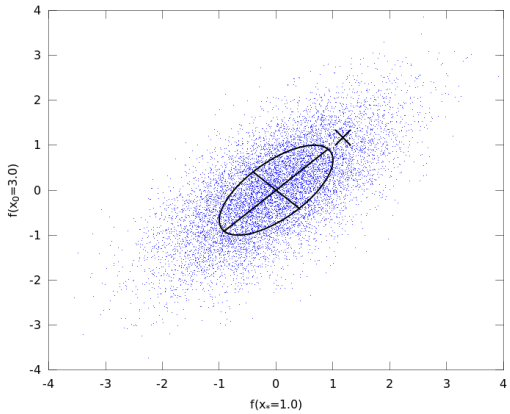
$$k_{se}(d) = e^{-a\cdot d^2}, \quad a \in \mathbb{R}^+, \quad d(s,t) = ||s-t||, \quad s,t \in \mathscr{S}. \tag{2.13}$$

A covariance function that is defined that way is called *stationary* and shares similarities with local learning whereas *non-stationary* kernels such linear or polynomial kernels implement global learning. In addition, kernels can be defined for periodic and symmetric functions.

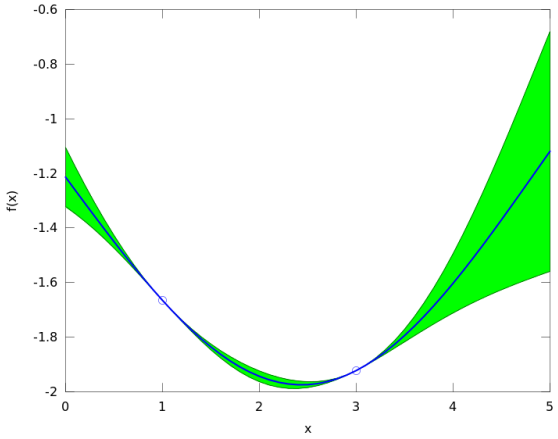
The mean function also allows expressing prior knowledge on the target function. Still, it is set to constant zero in nearly all applications. This is because often there is more knowledge about the relation between observations available than about how to make an absolute assumption. Further, absolute model knowledge can be directly included into the covariance where it can be more efficiently combined with noise models.



(a) The Gaussian process regression if only one training sample at $x_0 = 3.0$ has been observed.



(b) The bivariate distribution of function values given $S = \{x_0 = 3.0, x_\star = 1.0\}$. If the distribution is conditioned for an observed y_0, the most likely value for $f(x_\star)$ can be computed.



(c) Prediction of the latent function if two training samples are provided.

Figure 2.17.: Example of the squared exponential kernel.

Gaussian Process Regression (GPR)

Regression This paragraph explains the calculus necessary to make a prediction of an unknown output given its input and a set of previously sampled observations. shown on the illustrative example in Figure 2.17a using single training data point with $s_0 = 3.0$ and the squared exponential kernel with zero mean.

One can see that points that lie closer to the data point on the x-axis have a similar function value. They are closer to the x-axis otherwise because of the zero mean function. In order to evaluate a function f distributed by this Gaussian process at a given $x_\star$, the conditional distribution has to be observed. The index set is then

$$S = \{x_0 = 3.0, x_\star\}, \quad x_\star \in \mathbb{R}$$

and the covariance matrix of the bivariate Gaussian is

$$K_{x_\star} = \begin{bmatrix} 1 & k(x_\star, 3.0) \\ k(x_\star, 3.0) & 1 \end{bmatrix}.$$

Then, the conditioned distribution for an unknown $x_\star$ is (according to Eq. (2.10))

$$p(s_\star|s_0) \sim \mathcal{N}\left(\frac{k(s_\star,s_0)}{k(s_0,s_0)} \cdot f(s_0), k(s_\star,s_\star) - \frac{k(s_\star,s_0)^2}{k(s_0,s_0)}\right).$$

The actual prediction is the most likely value given $s_\star$ and s_0 which is exactly the mean value of the conditional distribution

$$f(s^\star) = \frac{k(s_\star,s_0)}{k(s_0,s_0)} \cdot f(s_0).$$

The variance, which is also shown in Fig. 2.17a, is a good measure for the confidentiality which decreases the farer one get away from the actual data point. The bivariate distribution for $x_\star = 1.0$ is shown in Fig. 2.17b. In the more general case of multiple data points, the conditional distribution becomes more complex to compute. This case is displayed for two data points in Fig. 2.17c. Again, it can be observed that the confidence decreases the further one gets away from the sample points, and the plot gives an idea of the smooth interpolation through data points. The general formula for the conditional distribution given an unknown position $x_\star$ and the n observed training samples $(x_0, f(x_0))$ up to $(x_n, f(x_n))$ is given by

$$p(s_\star|\boldsymbol{s}) = \mathcal{N}\left(\boldsymbol{k}_{s_\star} \cdot \bar{K}^{-1} \cdot \boldsymbol{y}, k(s_\star,s_\star) - \boldsymbol{k}_{s_\star} \cdot \bar{K}^{-1} \cdot \boldsymbol{k}_{s_\star}^t\right), \tag{2.14}$$

where (similar to Eq. (2.10))

$$K_{s_\star} := \begin{pmatrix} k(s_\star,s_\star) & \boldsymbol{k}_{s_\star} \\ \boldsymbol{k}_{s_\star}^t & \bar{K} \end{pmatrix}, \qquad \bar{K} := \begin{pmatrix} k(s_0,s_0) & \cdots & k(s_0,s_n) \\ \vdots & & \vdots \\ k(s_n,s_0) & \cdots & k(s_n,s_n) \end{pmatrix},$$

$$\boldsymbol{k}_{s_\star} =: (k(s_\star,s_0), \ldots, k(s_\star,s_n)), \qquad \boldsymbol{y} := (f(\boldsymbol{s}_0), \ldots, f(\boldsymbol{s}_n))^t$$

It is important to note that the matrix $\bar{K}^{-1}$ has to be calculated and inverted newly every time a *new sample* is added to the training data. This is the bottleneck in incremental and online learning as with increasing accuracy the computation time increases cubically. As a consequence, a sparsification of the data is required for learning in high-dimensional spaces (see Csató and Opper, 2002; Nguyen-Tuong et al., 2009).
A comprehensive introduction to GPR including applications in robotics is given in (Rasmussen and Williams, 2005). The approach of a body schema by Sturm et al. (2009) presented in the previous section also relied on GPR to learn the low-dimensional forward kinematics models in between observable

body parts. More recently, Hartmann et al. (2012) used multiple local Gaussian processes to learn the inverse dynamics for robot arm with seven degrees of freedom and artificial pneumatic muscles which cannot be modeled by classical methods. de la Cruz et al. (2012) recently presented an modification of the GPR (Csató and Opper, 2002) to learn the inverse dynamics of a simulated industrial robot with six degrees of freedom.

2.3.6. Decomposition

All methods presented in the previous subsections suffer from Bellman's "curse of dimensionality" (Bellman, 1960) when learning forward models with high-dimensional input spaces. When the number of input dimensions increases, the number of training samples required to learn the model grows exponentially. This means that no matter how efficient the learning algorithm is, the construction of a sensorimotor map that covers the whole configuration space becomes intractable on a humanoid robot with many degrees of freedom. Up to a certain degree, local methods can evade this problem. A single linear model requires only $\mathcal{O}(n)$ training examples for its construction. However, the number of local models itself grows exponentially when the complete configuration space has to be approximated.
To solve this problem, several methods have been proposed that attempt to decouple the input signals and decompose the learning problems into partial problems that can be learned more easily. Currently, such methods can only be applied to forward and inverse models of the kinematics. They impose restrictions that hinders the application on a humanoid robot that learns through self-observation. This section presents several approaches and how they can be applied for learning robot kinematics.

Function decomposition

Ruiz de Angulo and Torras (2005b) proposed a decomposition of the kinematics of robot manipulators with six controllable degrees of freedom with the additional restriction that the last three joint axes intersect in a single point. This restriction holds for the most industrial and anthropomorphic robot arms and allows decoupling the joint angles and therefore learning the inverse (and forward) kinematics more easily. Further, the *pose* of the end-effector, that is, its position and orientation, has to be observable by the sensors. This paragraph presents the decomposition using the notation used throughout this thesis. The function describing the kinematics is decomposed into two shorter partial inverse kinematic functions and two helper functions. At first, the joint angles $\boldsymbol{\theta}$ are separated into *proximal* and *lateral* joint angles $\boldsymbol{\theta}^p$ and $\boldsymbol{\theta}^l$, that is, the joint angles closer and more distal to the robot's base respectively

$$\begin{aligned} \boldsymbol{\theta} &= \big(\theta_1, \ldots, \theta_6\big), \\ \boldsymbol{\theta}^p &:= \big(\theta_1, \theta_2, \theta_3\big), \qquad & \boldsymbol{\theta}^l &:= \big(\theta_4, \theta_5, \theta_6\big). \end{aligned} \tag{2.15}$$

Then, the observations of the end-effector are separated into the position $\boldsymbol{x}$ and orientation X of the end-effector. Without loss of generality, the orientation will be expressed as a rotation matrix for simplicity.

The forward kinematics can then be decomposed into two functions—one for the position and one for the orientation

$$f_x(\cdot): \quad \Theta^3 \times \Theta^3 \to \mathbb{R}^3, \quad (\boldsymbol{\theta}^p, \boldsymbol{\theta}^l) \mapsto \boldsymbol{x}, \tag{2.16}$$

$$f_X(\cdot): \quad \Theta^3 \times \Theta^3 \to SO_3, \quad (\boldsymbol{\theta}^p, \boldsymbol{\theta}^l) \mapsto X. \tag{2.17}$$

Learning then uses a training set of the following form

$$\mathscr{T} = \left\{ \left(\boldsymbol{\theta}_i^p, \boldsymbol{\theta}_i^l, \boldsymbol{x}_i, X_i \right) : i = 1, \ldots, n, \boldsymbol{\theta}_i^p, \boldsymbol{\theta}_i^l \in \Theta^3, \boldsymbol{x} \in \mathbb{R}^3, X \in SO_3 \right\}$$

Definition The first of the partial inverse kinematics function maps the position of the *cross point* $\boldsymbol{x}_\star$—the point where the last three axes intersect—to the proximal joint angles

$$k_{\boldsymbol{x}_\star}(\cdot): \quad \mathbb{R}^3 \to \Theta^3, \quad \boldsymbol{x}_\star \mapsto \boldsymbol{\theta}^p \tag{2.18}$$

The first helper function $h_{\boldsymbol{x}_\star}$ maps the orientation of the end-effector to the translation from the end-effector position to the cross-point

$$h_{\boldsymbol{x}_\star}(\cdot): \quad SO_3 \to \mathbb{R}^3, \quad X \mapsto \boldsymbol{t} = \boldsymbol{x} - \boldsymbol{x}_\star, \tag{2.19}$$

such that

$$k_{\boldsymbol{x}_\star}\left(\boldsymbol{x} - h_{\boldsymbol{x}_\star}(X) \right) = f_x(\boldsymbol{x})^{-1} \tag{2.20}$$

The helper function $h_{\boldsymbol{x}_\star}$ can directly be determined by moving the last two joints which yields the last two joint axis and their intersections.

The second partial inverse kinematic function k_X maps the orientation to lateral joints $\boldsymbol{\theta}^l$ for a *fixed* set of proximal joint angles $\boldsymbol{\theta}_0^p$

$$k_X(\cdot): \quad SO_3 \to \Theta^3, \quad X \mapsto \boldsymbol{\theta}^l \tag{2.21}$$

It is accompanied by the second helper function h_X that maps proximal joints $\boldsymbol{\theta}^p$ to a rotation R

$$h_X(\cdot): \quad \Theta^3 \to SO_3, \quad \boldsymbol{\theta}^p \mapsto R = f_X(\boldsymbol{\theta}_0^p, \boldsymbol{\theta}^l) \cdot X^{-1}, \tag{2.22}$$

such that

$$k_X\left(f_X(\boldsymbol{\theta}_0^p, \boldsymbol{\theta}^l) \right) = k_X\left(h_X(\boldsymbol{\theta}^p) \cdot f_X(\boldsymbol{\theta}^p, \boldsymbol{\theta}^l) \right). \tag{2.23}$$

Each of the functions maps between spaces with lower dimensions such that the joint angles are effectively decoupled and learning requires much less training samples.

Learning and Application Ruiz de Angulo and Torras (2005b) proposed several schemes to learn the partial kinematics and helper functions with different degrees of parallelism. Here, only the algorithm that synchronously learns all functions will be presented. It is the only capable of learning online during the

Algorithm 2: Pseudo code the parallel refinement of the function decomposition by Ruiz de Angulo and Torras (2005b)

Input: Initialized estimators for h_X,$k_{\boldsymbol{x}_\star}$ and k_X

for $(\boldsymbol{\theta}_i^p, \boldsymbol{\theta}_i^l, \boldsymbol{x}_i, X_i) \in \mathscr{T}$ **do** `// For all samples`

  Adapt estimator for h_X to $\left(\boldsymbol{\theta}_i^p, k_X^{-1}(\boldsymbol{\theta}_i^l) \cdot X_i\right)$

  Adapt estimator for $k_{\boldsymbol{x}_\star}$ to $\left(x_i - h_{\boldsymbol{x}_\star}(X_i), \boldsymbol{\theta}_i^p\right)$

  Adapt estimator for k_X to $\left(h_X(\boldsymbol{\theta}_i^p) \cdot X_i, \boldsymbol{\theta}_i^l\right)$

normal application of the robot. For every sample, the functions are learned relying on the other's estimations. With the time, the estimators will converge to accurate approximations of the partial kinematics. The algorithm is shown in Algorithm. 2. The authors evaluated the decomposition with simple nearest-neighbor learning and parameterized self organized maps (see Section 2.3.3) on a simulated industrial robot.

This approach could successfully reduce the number of training samples required to learn the kinematic functions describing the robot. However, the restrictions to six degrees of freedoms and the necessity for the last axes to intersect is very exclusive and a more general learning approach is therefore desirable for the application on humanoid robots.

Decomposition into virtual robots

In a more recent work Ruiz de Angulo and Torras (2008) presented a decomposition of the forward kinematics that is less restrictive with respect to the kinematic structure of the robot, that is, it is not limited to six joint axes and the intersection of the last three axes in a single point is no more a requirement. However, it require that not only the pose of the end-effector is observable but also the pose of intermediate body parts, that is, markers attached to them. Then, a forward kinematics of the form

$$\begin{aligned} f(\cdot): \quad & \Theta^{d_{\text{in}}} \to SE_3 \\ & \boldsymbol{\theta} \mapsto t_{0,1}(\theta_1) \cdot t_{1,2}(\theta_2) \cdot \ldots \cdot t_{d_{\text{in}}-1,d_{\text{in}}}(\theta_{d_{\text{in}}}) = X, \end{aligned}$$

where the functions

$$t_{i-1,i}(\cdot): \quad \Theta \to SE_3,$$

are partial kinematics that map joint angles to rigid body transformations (homogeneous transformation matrices for instance) out of the special Euclidean group. They describe the transformation from one joint to its successor in dependence of the respective joint angle. A decomposition of the end-effector pose into position and orientation (as in the previous section) is not necessary.
The main idea is to decompose f such that partial kinematics that are observable can be learned separately. If the pose of the k-th body part X_k is observable for instance, f can be written as the product of two partial kinematics. Therefore the joint angles are divided into proximal and lateral joint angles (analogously to

Eq. (2.15))

$$\boldsymbol{\theta}^p := \left(\theta_1, \ldots, \theta_k\right), \qquad \boldsymbol{\theta}^l := \left(\theta_{k+1}, \ldots, \theta_{d_{\text{in}}}\right).$$

The partial forward kinematics are then

$$\begin{aligned} f_1\left(\boldsymbol{\theta}^p\right) &= t_{0,1}(\theta_1^p) \cdot t_{1,2}(\theta_2^p) \cdot \ldots \cdot t_{k-1,k}(\theta_k^p) = X_k \\ f_2\left(\boldsymbol{\theta}^l\right) &= t_{k,k+1}(\theta_1^l) \cdot t_{k+1,k+2}(\theta_2^l) \cdot \ldots \cdot t_{d_{\text{in}}-1,d_{\text{in}}}(\theta_{d_{\text{in}}-k}^l) = X \cdot X_k^{-1}. \end{aligned}$$

As a result, the joint angles are completely decoupled and can be learned *separately* and *simultaneously* with training sets

$$\begin{aligned} \mathscr{T}_1 &= \left\{ (\boldsymbol{\theta}_i^p, X_i) : i \in \{1, \ldots, n\},\ \boldsymbol{\theta}_i^p \in \Theta^k,\ X_i \in SE_3 \right\}, \\ \mathscr{T}_2 &= \left\{ (\boldsymbol{\theta}_i^l, X_i \cdot X_{k,i}^{-1}) : i \in \{1, \ldots, n, \},\ \boldsymbol{\theta}_i^l \in \Theta^{d_{\text{in}}-k},\ X_i, X_{k,i} \in SE_3 \right\}. \end{aligned}$$

Ruiz de Angulo and Torras (2008) evaluated the decomposition with a localized version of the parameterized self-organizing maps (see Section 2.3.3)

The effect of the decomposition is then biggest when $k \approx {}^{d_{\text{in}}}/_2$. Then the required number of training samples is reduced from $\mathscr{O}(\exp(d_{\text{in}}))$ to nearly its square $\mathscr{O}(\exp(d_{\text{in}}/2)$ which renders sensorimotor learning of kinematics with many degrees of freedom much more tractable. Further, the number of partial kinematics can be increased such that the number becomes linear in the number of inputs d_{in}. The decomposition then becomes similar to the body schema learning presented by Sturm et al. (2012) (see Section 2.2). The main drawback, however, is the increase of the number of observable body parts of the robot which have to be visible simultaneously during training. On a humanoid robot learning from self-observation with cameras mounted on the head, this is a restriction which often cannot be met. When the hand is close to the head, the field of vision is usually too small to observe more body parts than the end-effector. For that reason, the decompositions presented here were improved in this thesis to dispose of this necessity.

2.3.7. Discussion

In this last section, various methods from the field of machine learning were presented that are suitable for and have been applied to sensorimotor learning. They are able to create approximations of the complex non-linear relations between the different sensor and control modalities of a robot and map a high-dimensional input spaces (*proprioception*, i.e., joint angles and motor commands) to the usual lower-dimensional cartesian task space (kinematics) or force/torque space (dynamics).

The necessity for learning arises only if it becomes intractable to obtain sufficient knowledge about a robot's structure by classical means, for instance, due to discrepancies between design parameters and the actual robot or the lack of adequate measuring instruments and human intervention. Even if models can be determined, they are yet static and cannot account for dynamic changes in the robot's structure. Such changes can result from wear and tear but also from the interaction with the environment—most notably, the use of tools. Further, these problems usually coincide with higher numbers of controllable degrees of freedom. This also increases the complexity of the sensorimotor relations further impeding parameter identification.

Humanoid robots typically fulfill all these criteria: Due to energy and safety restrictions they are built more lightweight and less robust compared to their industrial counterparts and their anthropomorphic appearance and potentially likewise dexterity almost enforces the idea of tool use. Further, they usually come with a huge number of controllable degrees of freedom. Altogether, this qualifies them as the ideal target for sensorimotor learning.

The main criteria that allow a distinction of the presented methods into groups are the scope (global and local) and the nature of the model parameters (parametric and non-parametric) learning. Unfortunately, it is very difficult to directly compare and judge the members belonging to different groups as they all have evolved to be most efficient in their field of application. Furthermore, a fair comparison is almost always impossible due to different experimental conditions under which they were evaluated. That way, non-parametric and local learning has nearly always been used to learn mappings along trajectories—a reduction of the problem to one-dimensional subspaces embedded in the workspace. However, sensorimotor maps that represent a complete humanoid robot's body require learning of the whole reachable action space. In theory, all presented methods are capable of learning the complete region but their different paradigms render them differently qualified for this task. From this point of view, global learning seems to be the best for this task and the idea of the ability to generalize from few examples to the whole configuration space is intriguing. But the few global approaches presented within this section also require many training samples that have to be well distributed in the work space to build a globally accurate model.
It is an important observation that all presented methods act as general function approximations. Usually, this generality is an advantage as little to no knowledge of the target function is required in opposition to classical methods. The lack of a specialized inductive bias, however, is main reason that prevents rapid and exact global learning.

For this reason, the main contributions of this thesis are novel model representations for global parametric learning that exploit the nature of dynamic and kinematic forward models. The analytical functions that describe these models are composed of trigonometric functions. The new models and associated learning algorithms expose a bias to this type of functions and are consequently less general as the discussed approaches. Instead, they require significantly less training samples and generalize very well even to regions of the configuration space that were not explored during training (extrapolation). Sensor noise can reliably be compensated and under the perfect conditions in a simulation, the models are an exact representation of the underlying physical processes. Further, they are computationally tractable and can be learned incrementally such that they can be applied for online learning.
Bellman's 'curse of dimensionality' cannot be evaded however and the number of required training samples still grows exponentially in the number of controllable degrees of freedom. On the one hand, the decomposition techniques that were presented at the end of this section provide a simple method to overcome this problem. On the other hand, they are either limited to a subset of robots in their application or require perception that is not available on a humanoid robot. For this reason, a novel decomposition had to be developed that combines and extends the characteristics of the available techniques.

In the following three chapters, these contributions will be presented in detail along with the derivation of their mathematical foundation.

3. Representations for kinematic and dynamic models

The contributions of this thesis to the research in the field of body schema and sensorimotor learning in particular are presented in this and the following two chapters. They are mainly centered around a novel method for the representation of trigonometric sensorimotor relations and their related learning algorithms, which, above all, cover forward models of the kinematics and dynamics of robots with revolute joints. Because their principal use in kinematics and their mathematical roots in projective geometry these maps have been named *Kinematic Bézier Maps (KBM)* (Ulbrich et al., 2012c). The first section in this chapter provides a brief introduction to polynomial function approximation and projective geometry which both are fundamental for the definition of the KBM models. Section two consist of the detailed mathematical definition of these models, and their properties and relation to forward models. After that, in section three, their evolution to represent dynamic models is presented. The related algorithms for learning the proposed models will presented in the subsequent chapter four followed by the presentation of an enhanced variant of virtual robot decomposition in chapter five that addresses the problem of Bellman's "curse of dimensionality" (Bellman, 1960) in Chapter 5.

3.1. Introduction

The investigation of the state-of-the-art in body schema and particularly sensorimotor learning in chapter two showed that, although global model learning has superior characteristics for robot control, currently local linear techniques prevail due to the overwhelming complexity inherent in complex robot systems—especially in humanoid robots. This observation led to the emergence of novel methods for the global representation of sensorimotor relations designed for sensorimotor learning even in high-dimensional input spaces. The underlying mathematics is well-known in the domain of *computer aided geometric design (CAGD)* but have not been applied in robots in the context of sensorimotor learning so far. Sensorimotor maps inherently involve nonlinearities mostly in form of trigonometric relations to the input signals. A revolute joint, for instance, defines a trigonometric relation between its joint angle and the spatial location of the following arm element as it moves along a circular trajectory around the joint axis. Polynomials, for instance, cannot be used for creating an exact sensorimotor map and approximations require high-order polynomials, PSOM for instance (Klanke and Ritter, 2005), or have to be a composition of low-degree polynomials only with local validity such as LWPR in(Vijayakumar and Schaal, 2000), GMR in (Lopes and Damas, 2007), and XCSF in (Sicard et al., 2011).

In this chapter, it will be shown that the nature of the nonlinearities can also be exploited and forward models can be derived from *projective geometry*. Originally, this technique was designed to model just the

direct kinematics but it can be adopted to closely related sensorimotor maps as is shown for the inverse dynamics.

3.1.1. Polynomial function approximation

In this thesis, polynomial functions were investigated as possible candidates for global models applicable for sensorimotor learning at first. The only technique encountered that relies on higher-order polynomials encountered in the context of sensorimotor learning are the *Parameterized Self-Organizing Maps (PSOM)* (see Section 2.3.3). One property of function approximation with polynomials that simplifies learning is the linearity in the their parameters. Given two vector spaces $V^{d_{\text{in}}}$ and $V^{d_{\text{out}}}$ over the field V (the real numbers $\mathbb{R}$ for instance), a polynomial is defined as

$$p(\boldsymbol{x}) = \sum_{\boldsymbol{i} \in \mathscr{I}^n_{d_{\text{in}}}} \boldsymbol{p}_i \cdot P_{\boldsymbol{i}}(\boldsymbol{x}), \qquad P : V^{d_{\text{in}}} \to V^{d_{\text{out}}}, \quad \boldsymbol{x} \in V^{d_{\text{out}}}, \boldsymbol{p}_{\boldsymbol{i}} \in V^{d_{\text{out}}} \tag{3.1}$$

where $P_{\boldsymbol{i}}(\cdot)$ denotes the elements a polynomial basis of degree n in d_i input dimensions identified by index vectors $\boldsymbol{i}$ out of a set $\mathscr{I}^n_{d_{\text{in}}}$. The parameters $\boldsymbol{p}_{\boldsymbol{i}}$ then can be interpreted as points in the output space and appear only linear in this equation. For instance, the monomials

$$\mathscr{M}^2 = \{x^2y^2, x^2, y^2, x^2y, xy^2, xy, x, y, 1\}$$

form a basis for all quadratic polynomials (that is, $d = 2$ and $n = 2$) and can exemplarily be addressed by the degree of their variables

$$M_{\boldsymbol{i}=(i,j)} = x^i y^j, \quad \mathscr{I}^2_2 = \big\{(i,j) : i,j \in \{0,1,2\}\big\}.$$

Multivariate Lagrange interpolation

PSOM are representatives of self-organizing feature maps which in general do not divide training data into input and output signals during training (i.e., semi-supervised learning, see Section 2.3.3). The continuity introduced by the PSOM, however, requires this separation. The same learning rule as for Kohonen maps is applied and after learning, the weights of the neurons are interpolated by polynomials

$$p(\boldsymbol{\theta}) = \sum_{\boldsymbol{a} \in \mathscr{A}} \boldsymbol{p}_{\boldsymbol{a}} \cdot H_{\mathscr{A}}(\boldsymbol{a}, \boldsymbol{\theta}),$$

that uses a multivariate basis of the form

$$H_{\mathscr{A}}(\boldsymbol{a}, \boldsymbol{\theta}) = \prod_{j=1}^{d_{\text{in}}} L_{\mathscr{A}_j}(a_j, \theta_j)$$

with the Lagrange polynomials

$$L_{\mathscr{A}_\square}(\bar{a},\theta) = \prod_{\alpha\in\mathscr{A}_\square\setminus\{\bar{a}\}} \frac{\theta-a}{\bar{a}-a}.$$

The index set $\mathscr{A} = \mathscr{A}_i \times \ldots \times \mathscr{A}_{d_{\text{in}}}$ contains all possible coordinates of the neurons in the competitive layer. Using this base allows the parameters $\boldsymbol{p_a}$ of the resulting polynomial to be interpreted as points that are directly interpolated by the polynomial at $\boldsymbol{\theta} = \boldsymbol{a} \in \mathscr{A}$.

If not enough training data can be sampled for learning with the Kohonen rule, the parameters can directly be assigned to the output signals observed when sampling the input space in a regular grid such that the robot moves to each configuration $\boldsymbol{a} \in \mathscr{A}$. The observation of all configurations is problematic because of possible self-collision or occlusion. Further, the model over-fits to the noise in the output signals.

A simple alternative that completely discards the connection to self-organizing feature maps has been suggested without further investigation by Walter (1996). This approach exploits the linearity of the parameters in Eq. 3.1. The training with m observations in a training set

$$\mathscr{T} = \left\{(\boldsymbol{\theta}_j, \boldsymbol{x}_j) : j \in \{1,\ldots,m\}\right\} \tag{3.2}$$

yields optimal parameters $\boldsymbol{p}^\star_{\boldsymbol{a}}$ that can be determined such that they minimize the *mean least squares error (MSE)*

$$\boldsymbol{p}^\star_{\boldsymbol{a}} = \arg\min_{\boldsymbol{p_a}} \sum_j^m \left\| \left(\sum_{\boldsymbol{a}\in\mathscr{A}} \boldsymbol{p_a} \cdot H_{\mathscr{A}}(\boldsymbol{a},\boldsymbol{\theta}) \right) - \boldsymbol{x}_j \right\|^2, \qquad \boldsymbol{a} \in \mathscr{A}, m \geq \prod(n_j+1). \tag{3.3}$$

Algorithms for finding the optimal parameters are explained in Chapter 4 including online-applicable incremental versions for life-long learning. Due to the resulting simplicity of incrementally learning this polynomial model, it has been chosen for the experiments for the decomposition by Ulbrich et al. (2012b). The extension of the PSOM, called PSOM+, continues this path by not only optimizing the MSE but also regulating the curvature of the resulting polynomial manifold—considering all output dimensions. This results in a nonlinear optimization that also enforces the global character of the learning of polynomial models.

This demonstrates that global models can adequately be created with higher-order polynomial and robust algorithms minimizing the MSE. However, the choice of the polynomial order is important. The higher it is chosen, the more accurately the latent function can be approximated. This results also in a higher number model parameters which is directly linked to the volume of the training data. Thus a balance between accuracy and the duration of the training has to be found. In case of an anthropomorphic arm, it is nearly impossible to have a degree higher than two which already results (given seven degrees of freedom) in $3^7 = 2187$ samples per arm. While a quadratic or cubic interpolation may be adequate if only a small portion of the robots configuration space is relevant, it certainly is not for the complete space (see Fig. 3.1). This problem can only be overcome by the combination of locally valid polynomial models (cf., LWPR or local PSOM) or by a non-polynomial basis. This thought resulted in the continued research for a global model with the same benefits but maintain a lower level of complexity.

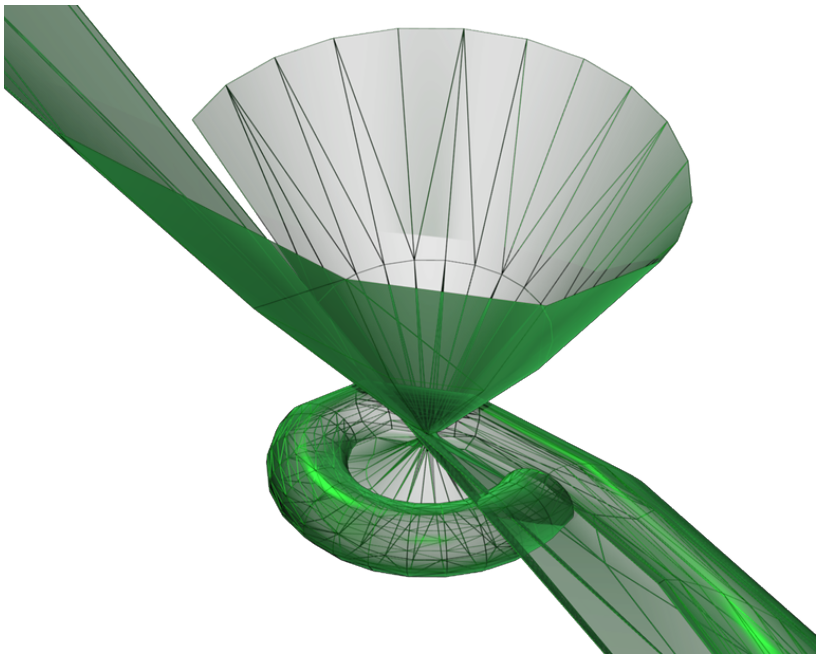


Figure 3.1.: Example of the approximation of the workspace of a robot with two orthogonal axes. The workspace is a torus. Ißt can be seen a segment of the torus is approximated very well (the area where training samples were taken from) but the extrapolation over the whole configuration space produces quickly large estimation errors because of the high polynomial degree of the Lagrangian basis.

Bézier approximation

There are many possible polynomial basis with different properties that can be considered for sensorimotor learning. The Bézier form will be of greater importance later in this chapter and is therefore introduced in brief[1]. The introduction in this paragraph is based on the detailed literature in (Prautzsch et al., 2002). It is an alternative polynomial representation based on a basis of Bernstein polynomials,

$$B_i^n(t) = \binom{n}{i} \cdot t^i \cdot (1-t)^{n-i}, \quad t \in \mathbb{R}. \tag{3.4}$$

A Bézier curve of degree n using this basis has the following form

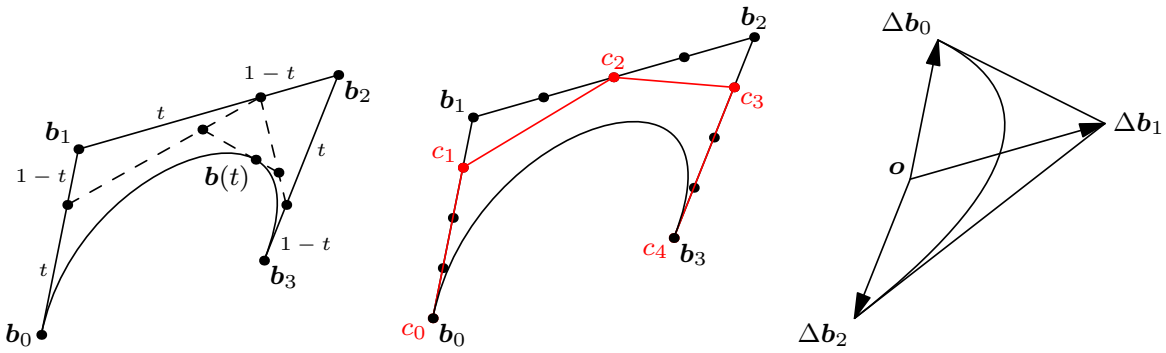
$$b(t) = \sum_{i=0}^{n} \boldsymbol{b}_i \cdot B_i^n(t). \tag{3.5}$$

For a given $t \in (0,1)$ the Bernstein polynomials $B_i^n(t)$ are positive and sum up to one

$$B_i^2(t) \in (0,1) \wedge \left(\sum_i^2 B_i^2(t) = (1-t+t)^n = 1 \right) : \forall t \in (0,1).$$

From this follows the important *convex hull property*—all points of the Bézier curve with parameters $t \in (0,1)$ lie in the convex hull spanned by the parameters $\boldsymbol{b}_i$, which are called *control points* or *control vertices*

[1] The Bézier form was developed in the automotive industry by Pierre Bézier at Renault and Paul de Casteljau at Citroën in the 1960ies



(a) A Bézier curve and DeCasteljau's algorithm to compute $\boldsymbol{b}(t)$.

(b) Degree elevation increases the polynomial degree of the curve.

(c) Derivatives are obtained as new Bézier curve by forward differences between the control points.

Figure 3.2.: An example of a cubic Bézier curve and important algorithms for its manipulation.

and form the *control polygon*. For $t = 0$ and $t = 1$ the first and the last basis polynomials, $B_0^n(0)$ and $B_1^n(1)$ respectively, equal one while all the others equal zero. This means that the first and the last control point both lie on the curve (*end-point interpolation property*).

DeCasteljau's algorithm (illustrated in Figure 3.2a) is a numerically stable method for the evaluation of a Bézier curve. It offers an descriptive geometric perspective on the evaluation of a function value given a parameter t. The n line segments connecting subsequent control points $\boldsymbol{b}_i$ and $\boldsymbol{b}_{i+1}$ are subdivided by the ratio $t : (1-t)$ and the procedure is repeated recursively with the resulting n points until a single point remains. The theory of Bézier curves is well developed and there are algorithms for *degree elevation* (raising the polynomial degree of a curve, see Figure 3.2b), smooth extrusion, and the calculation of derivatives (see Figure 3.2c) among others, many of which will be used in later sections. In order to represent multivariate polynomials, there exist two possibilities to modify the Bézier form. First, the Bernstein polynomials can be modified for multiple input variables. In the case of two inputs, this results in a triangular path with the polynomial basis

$$B_{i,j,k}^n(u,v,w) = \frac{n!}{i!j!k!} u^i v^j w^k, \quad u,v,w \in \mathbb{R},\ i,j,k \leq 0 \wedge i+j+k = n,$$

where u,v and w are called barycentric coordinates. This way, one can represent surfaces that are of the same polynomial degree n in any direction Δt. The second—and for this work more relevant approach—works similar to the multivariate Lagrange polynomials used for the PSOM. It is called *tensor-product representation*. It assumes that the resulting polynomial is necessarily of degree n *only* in its main directions, that is when only one of the input parameters changes. The resulting form for a bi-quadratic surface is

$$\boldsymbol{b}(s,t) = \sum_i^n \underbrace{\left(\sum_j^n \boldsymbol{b}_{i,j} B_j^n(s) \right)}_{\boldsymbol{b}_i(s)} B_i^n(t).$$

One can observe that this equation can be decomposed into two univariate Bézier curves which determine

each others control points. For more inputs we adapt the formalism

$$\boldsymbol{b}(\boldsymbol{t} = (t_1, \ldots, t_d)) = \sum_{\boldsymbol{i} \in \mathscr{I}^n_{d_{\text{in}}}} \boldsymbol{b_i} \cdot \prod_j^{d_{\text{in}}} B_j(t_{i_j}) = \sum_{\boldsymbol{i} \in \mathscr{I}^n_{d_{\text{in}}}} \boldsymbol{b_i} \cdot B_{\boldsymbol{i}}(\boldsymbol{t}), \tag{3.6}$$

with the shorter notation

$$B_{\boldsymbol{i}}(\boldsymbol{t}) := \prod_j^{d_{\text{in}}} B_j(t_{i_j}) \tag{3.7}$$

where the index set $\mathscr{I}^n_{d_{\text{in}}}$ is defined as

$$\mathscr{I}^n_{d_{\text{in}}} := \left\{ (i_1, \ldots, i_{d_{\text{in}}}) : i_j \in \{0, \ldots, n\} \forall j \in \{1, \ldots, d_{\text{in}}\} \right\}. \tag{3.8}$$

The multivariate forms of the Bézier representation share many of the properties of the univariate case including end-point interpolation, convex property, and also the algorithms for derivatives, degree elevation and evaluation. The algorithms are numerically stable because of the exclusive application of convex operations.

3.1.2. Primer on projective Geometry

The search in this thesis for models that can represent sensorimotor maps and are linear in their parameters led to rational polynomials. The derivation of these models requires a brief introduction to projective geometry, projections and the projective space. Detailed information on this topic can be found in (Farin, 1999).

A projective space $\mathscr{P}^d$ is closely related to its affine space $\mathscr{A}^d$ and the Euclidean space $\mathbb{R}^d$. An affine space is an extension of the Euclidean space that contains points and vectors expressed in *extended* or *homogeneous* coordinates which will be indicated by hollow small letters

$$\mathbb{x} = \begin{pmatrix} \boldsymbol{x} \\ h \end{pmatrix}, \quad h \in \{0, 1\}, \tag{3.9}$$

such that

$\mathbb{a} = \begin{pmatrix} \boldsymbol{a} \\ 1 \end{pmatrix}$ if $\boldsymbol{a}$ is a point and

$\mathbb{v} = \begin{pmatrix} \boldsymbol{v} \\ 0 \end{pmatrix}$ if $\boldsymbol{v}$ represents a direction vector (the difference between two points).

That way, *affine transformations* can be defined such that linear transformations in the Euclidean space (e.g., rotation, scaling and shear around the origin) can be combined with translations and be expressed in

Figure 3.3.: Raffael, *La scuola di Atene*, 1509–1510, Stanza della Segnatura, Vatican city Paralell lines that point away from the observer intersect in a *vanishing point* in the center of the painting. *(Source: Wikipedia)*

homogeneous coordinates

$$\boldsymbol{b} = A\boldsymbol{x} + \boldsymbol{t}$$
$$\Longleftrightarrow$$
$$\mathbb{b} = \begin{pmatrix} \boldsymbol{b} \\ 1 \end{pmatrix} = \begin{pmatrix} A & \boldsymbol{o} \\ \boldsymbol{o}^T & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix} + \begin{pmatrix} \boldsymbol{t} \\ 0 \end{pmatrix} = \begin{pmatrix} A & \boldsymbol{t} \\ \boldsymbol{o}^T & 1 \end{pmatrix} \begin{pmatrix} \boldsymbol{x} \\ 1 \end{pmatrix},$$

where $\boldsymbol{o}$ here denotes the null vector. This enables, for instance, to formulate rotations around arbitrary axis in the affine space.

Homogeneous coordinates are also used to describe the $d+1$-dimensional elements of a projective space $\mathscr{P}^d$. It is defined by its elements $\mathbb{p}$ which are the rays starting from the projective center $\mathbb{o} = (\boldsymbol{o}, 0)^T$ (which itself is not included in $\mathscr{P}^d$). The default affine space $\mathscr{A}^d$ itself is a subspace of $\mathscr{P}^d$ defined by intersection with a hyperplane. If not stated otherwise this hyperplane is defined by the equation $h = 1$. If a ray $\mathbb{p}$ intersects $\mathscr{A}^d$ in the point $\boldsymbol{p}$ (when γ equals one),

$$\mathbb{p} = \gamma \cdot \begin{pmatrix} \boldsymbol{p} \\ 1 \end{pmatrix}, \qquad \gamma \in \mathbb{R}_{\neq 0}, \tag{3.10}$$

$\boldsymbol{p}$ is called the *affine image* of $\mathbb{p}$. If a ray $\mathbb{f}$ is parallel to the affine hyperplane (i.e., they intersect for $\gamma = \infty$),

$$\mathbb{f} = \gamma \cdot \begin{pmatrix} \boldsymbol{f} \\ 0 \end{pmatrix}, \qquad \gamma \in \mathbb{R}_{\neq 0}, \tag{3.11}$$

it is called a *far point* or *direction*. The factor γ is called the *(projective) weight* of the point $\mathbb{p}$ or $\mathbb{f}$. The projective space does not only include the affine space but also all its possible images resulting from perspective projections each lying on a different hyperplane in $\mathscr{P}^d$ that does not contain $\mathbb{o}$. Analogously to affine transformations, a projection π is expressed by a square matrix. It is defined by $d+1$ projective

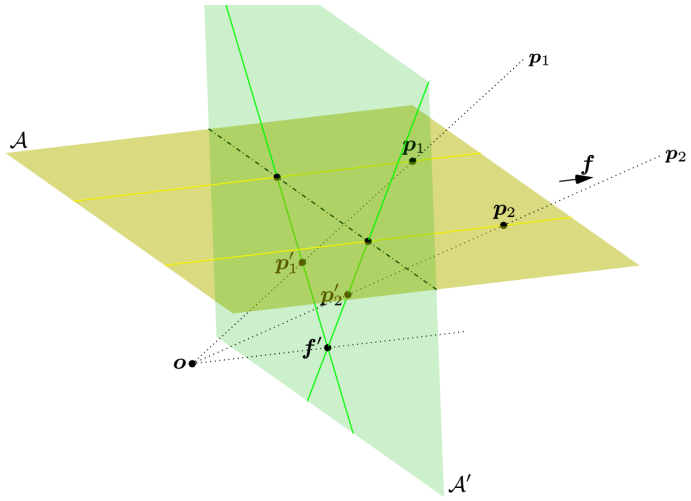


Figure 3.4.: Example of a perspective projection that maps two parallel lines through $\boldsymbol{p}_1$ and $\boldsymbol{p}_2$ onto another affine space $\mathscr{A}'$. After projections the two lines intersect in $\boldsymbol{f}'$ (cf. the vanishing point in Figure 3.3) which is a far point in the original $\mathscr{A}$.

points $\mathbb{p}_i$ and their images $\mathbb{q}_i$

$$\begin{aligned}\pi : \mathbb{x} \mapsto A \cdot \mathbb{x}, \quad A &= (\mathbb{q}_1 \dots \mathbb{q}_{d+1}) \cdot (\mathbb{p}_1 \dots \mathbb{p}_{d+1})^{-1} \\ &= (\gamma_1 \boldsymbol{q}_1 \dots \gamma_{d+1} \boldsymbol{q}_{d+1}) \cdot (\rho_1 \boldsymbol{p}_1 \dots \rho_{d+1} \boldsymbol{p}_{d+1})^{-1}\end{aligned}$$

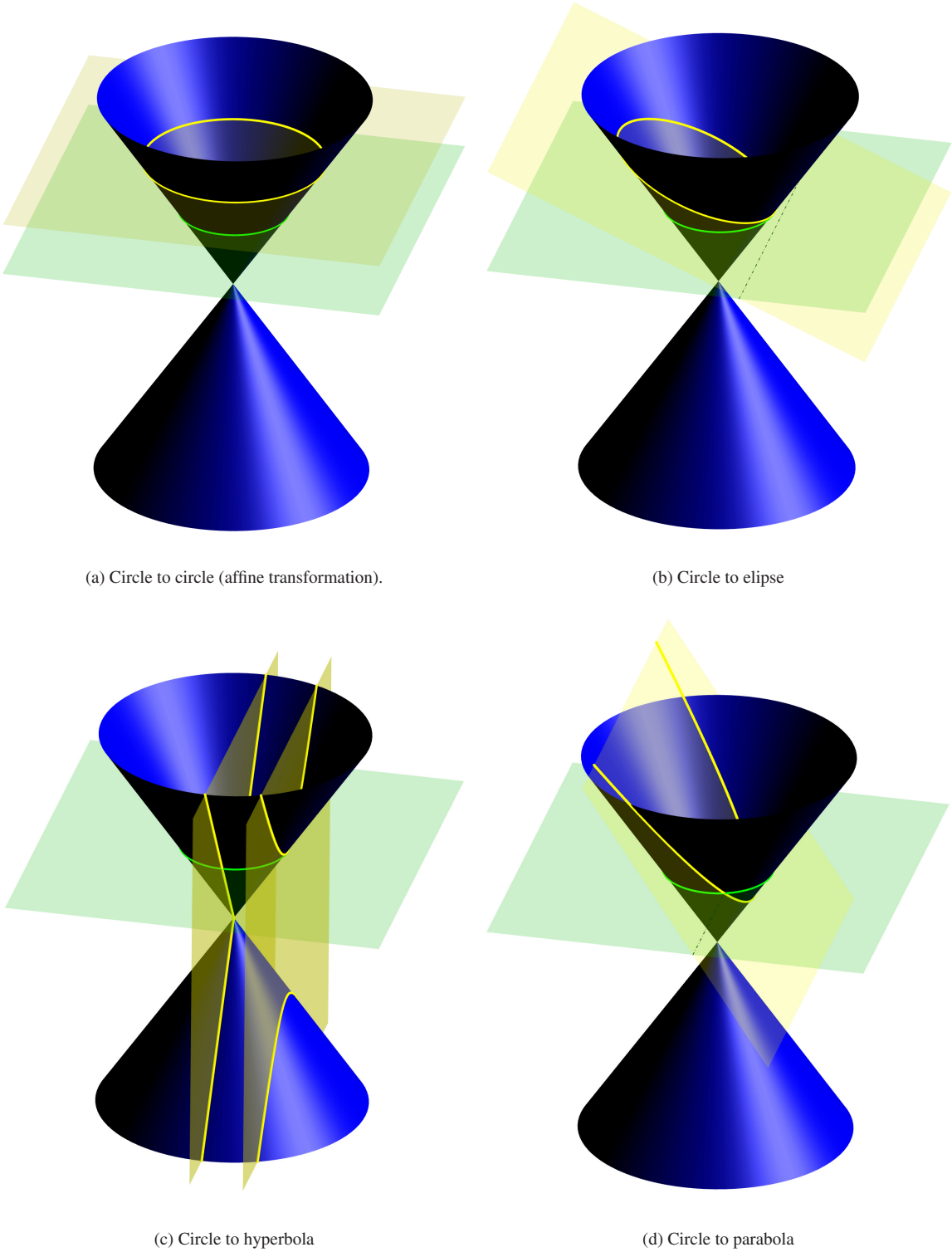
However, the projective weights γ_i and ρ_i have to be well defined in order to obtain a unambiguous projection. Therefore, an additional point $\mathbb{p}_{d+1}$ and its image $\mathbb{q}_{d+2}$ are required such that

$$\rho_1 \boldsymbol{p}_1 + \dots + \rho_{d+1} \boldsymbol{p}_{d+1} = \boldsymbol{p}_{d+2}, \quad \text{and} \quad \gamma_1 \boldsymbol{q}_1 + \dots + \gamma_{d+1} \boldsymbol{q}_{d+1} = \boldsymbol{q}_{d+2}.$$

Unlike affine transformations, such projections do not in general preserve ratios or parallel lines but—and this is most interesting in the following—can map far points onto an affine hyperplane, for instance, the intersection between parallel lines (see vanishing points in perspective drawing in Figure 3.3 and the geometric construction of a vanishing point in Figure 3.4).

Under projection, properties of many geometric objects alter significantly. For this work, it is most important how circles (and conics in general) behave as they represent the trigonometric functions encountered in sensorimotor learning.

A circle in the affine plane $\mathscr{A}^2$ can be illustrated in $\mathscr{P}^2$ as in Figure 3.5 without loss of generality. The points that belong to circle are the affine images of projective rays that themselves lie on the lateral surface of a double cone. A perspective projection defines a hyperplane that intersects the projective space and the cone resulting in another conic (see Figure 3.5a). It directly becomes clear that a projection can map the circle onto any other conic (see Figure 3.5b–3.5d) including the parabola where one point of the circle becomes a far point. This holds the key to the representation of circular movement that has been searched for in this thesis: Parabolas are low-degree polynomials and the examples shown that, given the projection



(a) Circle to circle (affine transformation).

(b) Circle to elipse

(c) Circle to hyperbola

(d) Circle to parabola

Figure 3.5.: Illustration of a circle in a two-dimensional affine space. The projective pre-images of the affine points that lie on the circle form a double cone. A projective projections is equivalent to the intersection of this cone with a different hyperplane. That way the circle can be mapped to any type of conic.

is known, it induces a quadratic parameterization $\mathfrak{c}(t)$ onto the circle

$$\mathfrak{c}(t) = A \cdot \begin{pmatrix} t^2 \\ t \\ 1 \end{pmatrix}, \tag{3.12}$$

where A denotes a projection and $(t^2, t, 1)^T$ the normal parabola. Later in this thesis, it will be show that, under a fixed projection, parameters of a parabola can be learned that maps onto the circle and how to combine subsequent rotations into a single multivariate *rational* polynomial.

3.1.3. Rational Bézier form

Figure 3.5d gives an impression on how the affine hyperplanes have to be defined in order to map the parabola in Eq. (3.12) onto a conic, that is, to induce a rational parametrization to the circle (Farin, 1999). Given the Bézier form in a projective space

$$\mathfrak{b}(t) = \sum_{i=0}^{2} \mathfrak{b} \cdot B_i^2(t) = \sum_{i=0}^{2} \gamma_i \cdot \boldsymbol{b}_i \cdot B_i^2(t) = \gamma(t) \cdot b(t),$$

the parabola is defined by its weights γ_i such that its affine image (i.e., its projection onto the default affine space $\mathscr{A} : h = 1$) lies on the conic. This projection has the form

$$b(t) = \frac{\sum_{i=0}^{2} \gamma_i \cdot \boldsymbol{b}_i \cdot B_i^2(t)}{\sum_{i=0}^{2} \gamma_i \cdot B_i^2(t)}.$$

The devision results in points on the curve $b(t)$ in the default affine plane, that is with $\gamma(t) = 1$, and is the reason for which the form is called *rational*. The shape of the control polygon and its weights determine the relation between the hyperplanes and hence the appearance of the resulting curve. For instance, a bigger weight γ_1 decreases the distance of the control point $\boldsymbol{b}_1$ to the curve, hence the name *weight* in reference to gravity. By the choice of the weights γ_i, the affine hyperplane the parabola resides in and its relation to the default affine space $\mathscr{A}$ is defined.

The following rules apply for the rational parametrization of conics: Given a control polygon with the points $\boldsymbol{b}_0$, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ that form an isosceles triangle and the weights γ_0 and γ_2 equal one, then the resulting curve is

- a hyperbola if $\gamma_1 > 1$,
- an ellipse if $0 < \gamma_1 \leq 1$ and
- a circle if $\gamma_1 = \cos(\alpha)$, when α is the isosceles common angle at $\boldsymbol{b}_0$ and $\boldsymbol{b}_2$.

An outline for the proof of the last rule is given in Figure 3.6. The following section will show how these observations lead to a model of kinematics that can be used for machine learning. For the representation with polynomials that map onto circular arcs, the Bézier form has been chosen. This is, on the one hand, due to its beneficial geometric properties, for instance, the convex property and its numeric stability. On the other hand, it is possible to formulate the conditions necessary for an implicit perspective projection that results in the transformation of the parabola into the desired circular arc.

3.2. Kinematic Bézier Maps: a rational parameterized model for kinematics

This section presents the *Kinematic Bézier Maps (KBM)* which are a class of parametric models that can be used to exactly represent the kinematics of robots with revolute joints such as those applied on humanoid robots. It can efficiently be learned due to the linearity in its model parameters. The KBM were previously presented in (Ulbrich et al., 2009, 2012c) and are the main contribution of this thesis to the field of machine learning and robotics. They are based on the observation that the kinematics can be described by a manifold with circular arcs in its main directions as it is the product space of subsequent rotations.

3.2.1. Formulation

Parameter transformation

Previously, it has been demonstrated how and under which conditions a parabola can be projected onto a circle, that is how a circular arc can be rationally parameterized. The input parameters of a parabola

$$p(\cdot) : (-\infty, \infty) \rightarrow \mathbb{R}^2, \tag{3.13}$$

however, do not coincide with the generating angles of a circle

$$c(\cdot) : (-\pi, \pi) \rightarrow \mathbb{R}^2. \tag{3.14}$$

For this reason, the joint angles have to be first transformed into the correct parameters required for the rational parametrization. This can be achieved with the following parameter transformation

$$\tau_\alpha(\theta) = \frac{\tan(\theta/2)}{2 \cdot \tan(\alpha/2)} + 1/2, \tag{3.15}$$

where α is again the common angle of the isosceles control triangle (see Figure 3.6). The graph of this transformation is shown in Figure 3.7. The transformation includes the 'replacement of the range half-angle', a substation that is common in algebraic analysis of kinematics (Kovács and Hommel, 1993). For a

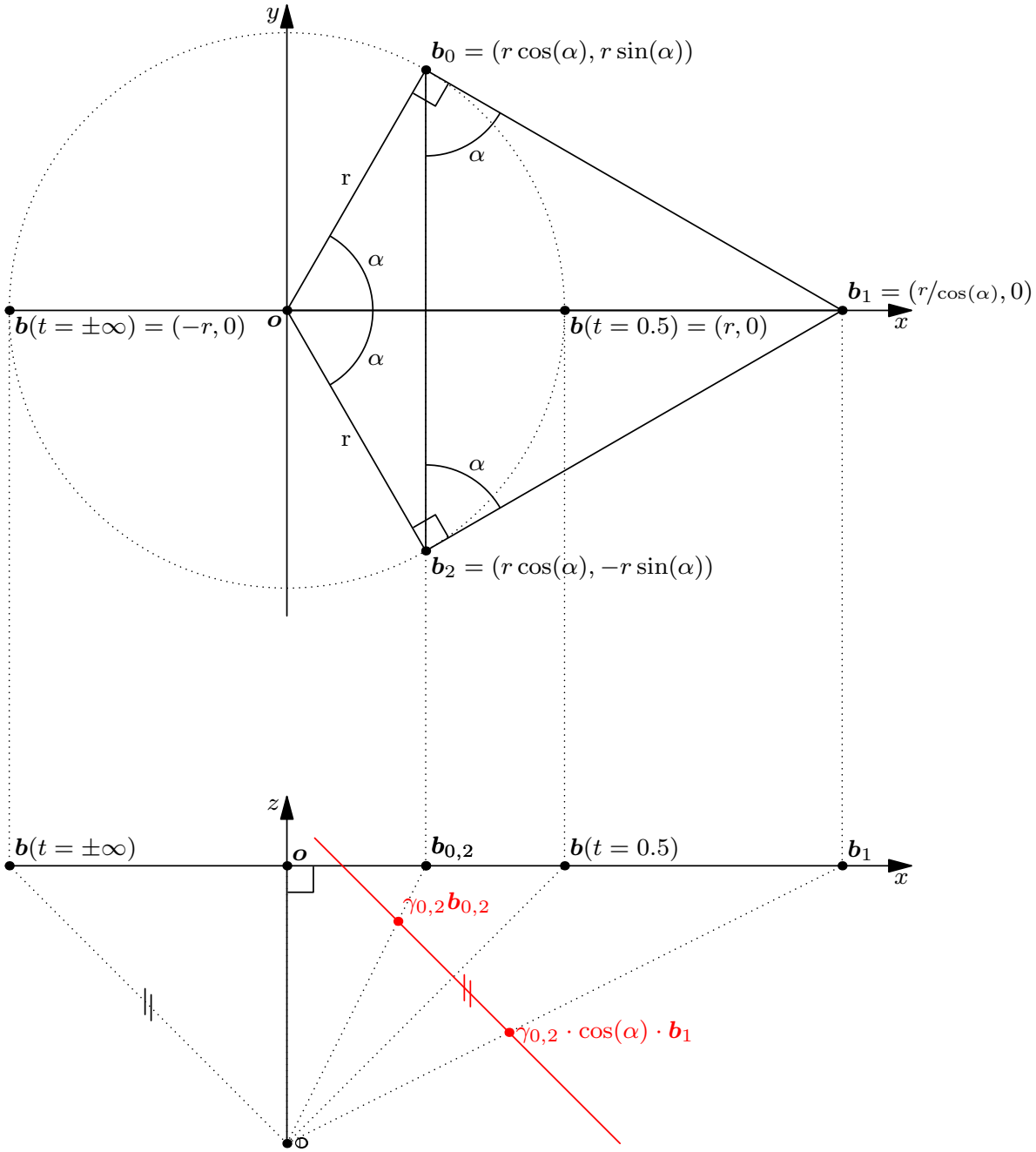
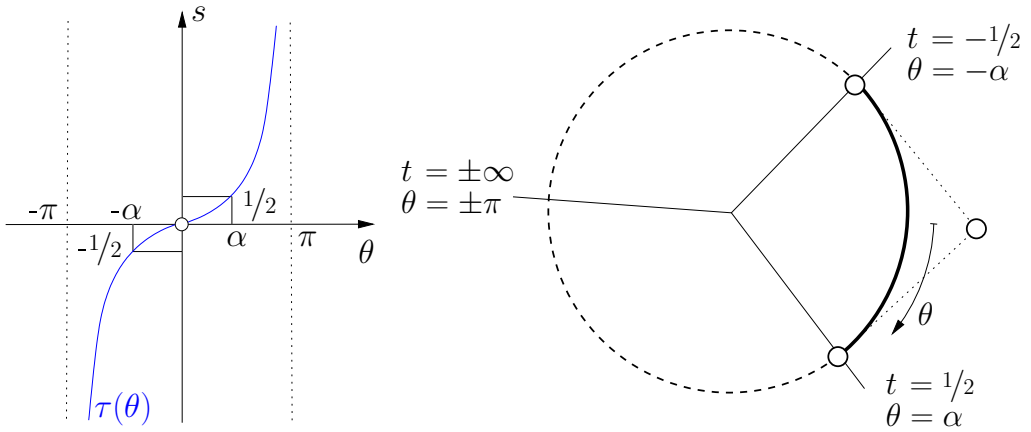


Figure 3.6.: Outline for the proof of the conditions required to represent a circular arc with a rational Bézier curve. Above in the image, the affine plane is shown while a side view is depicted below where the affine plane collapses to a straight line (the x-axis). Because of the circles symmetry, the triangle formed by $\boldsymbol{b}_0$, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ has to be an isosceles triangle. The parabola to be mapped onto the circle has to lie in an affine space (another hyperplane in $\mathscr{P}^2$ *(red)*) that is parallel to the one of the cone's enclosing rays (see Figure 3.5d). From the intersection of $\mathbb{b}_2$ such an affine hyperplane follows the constraint that $\gamma_0 = \gamma_2$ and $\gamma_1 = \cos(\alpha) \cdot \gamma_0$.



(a) Tangent half-angle substitution.

(b) Angles and their corresponding parabola parameters.

Figure 3.7.: Transformation between the parameters of the parabola and the circle. *Source: (Ulbrich et al., 2012c)*

proof, a rational Bézier parameterization of the unit circle will be solved for its parameter t without loss of generality

$$\Big(\cos\theta,\sin\theta\Big)^T \equiv b(t) = \Big(b_1(t),b_2(t)\Big)^T = \begin{pmatrix} \frac{4\cdot t^2\cdot\cos\alpha - 4\cdot t^2 + \cos\alpha + 1}{-4\cdot t^2\cdot\cos\alpha + 4\cdot t^2 + \cos\alpha + 1} \\ \frac{4\cdot t\sin\alpha}{-4\cdot t^2\cdot\cos\alpha + 4\cdot t^2 + \cos\alpha + 1} \end{pmatrix}.$$

Insertion into the identity

$$\tan(\theta/2) = \frac{\sin\theta}{1+\cos\theta} = \frac{b_2(t)}{1+b_1(t)} = 2t\cdot\frac{\sin\alpha}{\cos\alpha+1} = 2\cdot t\cdot\tan(\frac{\alpha}{2}),$$

yields

$$t = \frac{\tan(\theta/2)}{2\cdot\tan(\alpha/2)} \quad \square$$

Tensor-product form

The previous paragraph showed how the movement of the end-effector caused by moving a single joint can be described by a rational polynomial with known fixed weights. The next step towards the definition of the KBM model is to include multiple joints in the equation. Possible candidates for the integration are *multivariate rational Bézier functions* and *the tensor product form*. Quadratic multivariate Bézier curves (see Section 3.1.1) consist of fewer model parameters, that is, control vertices in the parameter set

$$|\mathscr{C}_{\text{multivariate}}| = \frac{(d_{\text{in}}+1)\cdot(d_{\text{in}}+2)}{2} \in \mathscr{O}(n^2),$$

which can be proven by induction. At a first glance, this smaller number makes them appear more suitable for learning the kinematic relations. However, the functions represented in such a Bézier form are *quadrics*, that is, every linear joint movement $\Delta\boldsymbol{\theta}$ results in a movement of the end-effector along a quadratic curve. A negative example demonstrating that this does not hold in general can be constructed easily. The workspace of a robot with two orthogonal joint axes is shaped like a torus, and any linear movement in the configuration space $\Delta\theta$ results only then in a quadratic movement in Cartesian space, if and only if it is parallel to one of

the unity vectors (i.e., only one joint moves at a time). In general, the TCP moves on a quartic curve which is displayed in Figure 3.8. This forward kinematics consequently is bi-quadratic and only the tensor product form can describe this kind of manifold.

The tensor product form consists of an amount of control vertices exponential in the number of active degrees of freedom

$$|\mathscr{C}_{\text{tensor}}| = 3^{d_{\text{in}}} \in \mathscr{O}(e^{d_{\text{in}}})$$

and each vertex belongs to d_{in} univariate curves related to a single degree of freedom. They can be identified by an index vector $\boldsymbol{i}$ out an index set $\mathscr{I}_n^{d_{\text{in}}}$ defined as

$$\mathscr{I}_n^{d_{\text{in}}} = \left\{ \boldsymbol{i} = (i_1, \ldots, i_{d_{\text{in}}}) : i_j \leq n \; \forall j \in \{1, \ldots, d_{\text{in}}\} \right\}, \tag{3.16}$$

where the lower index n always equals two except for learning the inverse dynamics (see Section 3.2.2). Additionally, two helper functions ι and $\bar{\iota}$ are defined to map these indices to natural numbers, for instance, to be able to arrange the control vertices in the parameter set $\mathscr{C}$ into the columns of a matrix C.
The invertible transformation ι assigns a natural number to each of the $n^{d_{\text{in}}}$ indices in $\mathscr{I}_n^{d_{\text{in}}}$

$$\iota_n^{d_{\text{in}}}(\cdot): \quad \mathscr{I}_n^{d_{\text{in}}} \to \mathbb{N}^+, \quad \boldsymbol{i} \mapsto i_1 \cdot n^0 + i_2 \cdot n^1 + \ldots + i_{d_{\text{in}}} \cdot n^{d_{\text{in}}-1} \tag{3.17}$$

If unambiguous in the current context, the indices of this function will be omitted. A matrix C can then be constructed out of the control points in $\mathscr{C}$ such that the control vertices are stored in its rows

$$C = \left(b_{\iota^{-1}(j),i} \right)_{j,j} \in \mathbb{R}^{3^{d_{\text{in}}} \times d_{\text{out}}}, \quad \boldsymbol{b_i} \in \mathscr{C}. \tag{3.18}$$

The second helper function, $\bar{\iota}$, maps every control vertex to the vertices of the triangle with respect to a specified degree of freedom associated to θ_j, that is, to its neighboring vertices and itself.

$$\begin{aligned} \bar{\iota}_n^{d_{\text{in}}} &: \mathscr{I} \times \{1, \ldots, d_{\text{in}}\} \to \mathbb{N}^3 \\ (\boldsymbol{i}, j) &\mapsto \begin{pmatrix} \iota_n^{d_{\text{in}}}\left((i_1, \ldots, i_{j-1}, 0, i_{j+1}, \ldots, i_{d_{\text{in}}})\right) \\ \iota_n^{d_{\text{in}}}\left((i_1, \ldots, i_{j-1}, 1, i_{j+1}, \ldots, i_{d_{\text{in}}})\right) \\ \iota_n^{d_{\text{in}}}\left((i_1, \ldots, i_{j-1}, 2, i_{j+1}, \ldots, i_{d_{\text{in}}})\right) \end{pmatrix}. \end{aligned}$$

This helper function is useful for the analysis of the symmetry in the control net (see Section 4.4) and also for the plotting of the control net (see Figure 3.9). Both ι^{-1} and $\bar{\iota}$ are evaluated frequently such that it is advisable to store their values in lookup tables for faster computation. Pseudo-code implementations for the creation of the look up tables are given in Algorithm. 4 and Algorithm. 3 respectively.

Now, the tensor product of the rational Bézier form can be finally defined as

$$b(\boldsymbol{\theta}) := \frac{\sum_{\boldsymbol{i} \in \mathscr{I}_2^{d_{\text{in}}}} \boldsymbol{b_i} \cdot \gamma_{\boldsymbol{i}} \cdot (B_{\boldsymbol{i}} \circ \tau_\alpha)(\boldsymbol{\theta})}{\sum_{\boldsymbol{i} \in \mathscr{I}_2^{d_{\text{in}}}} \gamma_{\boldsymbol{i}} \cdot (B_{\boldsymbol{i}} \circ \tau_\alpha)(\boldsymbol{\theta})} \tag{3.19}$$

Algorithm 3: Pseudo code for the creation of a lookup table for the reverse index function ι^{-1}

Input: Index function $\iota_3^{d_{\text{in}}}, \iota_3^{d_{\text{in}}} : \mathbb{N}^+ \to \mathscr{I}^{d_i}$
Output: Lookup table I for $\iota^{-1} \in \mathbb{R}^{3^{d_{\text{in}}} \times d_{\text{in}}}$

for $i = 1 : 3^{d_{in}}$ **do** // For all control vertives
  $r_1 := i - 1$
  for $j = 1 : d_{in}$ **do**
    $I_{i,j} := r_j \mod 3$ // store the index i_j in the lookup table
    $r_{j+1} := r_j - (r_j \mod 3)$

Algorithm 4: Pseudo code for the creation of a lookup table for $\bar{\iota}$.

Input: Index function $\iota_3^{d_{\text{in}}}, \iota_3^{d_{\text{in}}} : \mathbb{N}^+ \to \mathscr{I}^{d_i}$
Output: Lookup table $\bar{I}$ for $\bar{\iota}$, $\bar{I} \in \mathbb{R}^{3^{d_{\text{in}}} \times d_{\text{in}}}$

for $i = 1 : 3^{d_i}$ **do** // For all control vertives
  for $j = 1 : d_{in}$ **do** // For all degrees of freedom
    for $k = 1 : 3$ **do**
      $\boldsymbol{i} = (i_1, \ldots, i_{d_{\text{in}}}) := \iota^{-1}(j)$
      $\boldsymbol{w} := (w_1, \ldots w_{d_{\text{in}}}), \quad w_l = \begin{cases} i_l, & l \neq j \\ k, & l = j \end{cases}$ // get the neighboring vertices
      $\bar{I}_{i,j,k} = \iota(\boldsymbol{w})$ // store the index in the lookup table

using the simplified notation

$$(B_{\boldsymbol{i}} \circ \tau_\alpha)(\boldsymbol{\theta}) = \prod_j^{d_{\text{in}}} B_{i_j}^2(\tau_\alpha(\theta_j))$$

where the $\boldsymbol{b_i}$ are the $3^{d_{\text{in}}}$ control vertices and the $\gamma_{\boldsymbol{i}}$ the respective weights which still have to be determined for the multivariate case.

Analogously to the one-dimensional case, the weights γ_i can be determined using the a priori knowledge of the circular conditions in Section 3.1.3. The derivation will be shown at the example of two degrees of freedom for simplicity but can easily be applied to an arbitrary number d_{in}. Starting with modified formulation of the tensor product

$$b\Big((\theta_1, \theta_2)\Big) \cdot \bar{\gamma} = \sum_{i=0}^{2} \underbrace{\Big(\sum_{j=0}^{2} \boldsymbol{b}_{(i,j)} \cdot \omega_{1,(i,j)} \cdot B_j^2(\tau_\alpha(\theta_1)) \Big)}_{\bar{b}_i(\theta_1)} \cdot \omega_{2,(i)} \cdot B_i^2(\tau_\alpha(\theta_2)), \tag{3.20}$$

where the denominator in Eq. (3.19) is abbreviated by $\bar{\gamma}$ and the weights are defined as a product

$$\gamma_{i,j} = \omega_{1,(i,j)} \cdot \omega_{2,(i)}.$$

If θ_1 remains at a fixed position $\theta_1^\star$, the end effector moves on a circular curve $b_{\theta_1^\star}(\theta_2)$. Consequently, it

Algorithm 5: Pseudo code the evaluation of a KBM model

Input: control vertices $\mathscr{C}$
Output: Lookup table $\bar{I}$ for $\bar{\imath}$, $\bar{I} \in \mathbb{R}^{3^{d_{\text{in}}} \times d_{\text{in}}}$

Function *KBM(joint angles* $\boldsymbol{\theta}$*)*
  $B = \left(0\right)_{i,j} \in \mathbb{R}^{3^{d_{\text{in}}},1}$ `// Initialize basis polynomials`
  $C = \left(\boldsymbol{b}_{\imath^{-1}(i),j}\right)_{i,j} \in \mathbb{R}^{3^{d_{\text{in}}},1}, \quad \boldsymbol{b_i} \in \mathscr{C}$ `// compose matrix with control vertices`
  for $i = 1 : 3^{d_i}$ **do** `// For all control vertives`
    for $j = 1 : d_{in}$ **do** `// For all degrees of freedom`
      $\boldsymbol{i} = (i_1, \ldots, i_{d_{\text{in}}}) := \imath^{-1}(j)$
      $b_i := b_i \cdot \binom{2}{i_j} \cdot \tau_\alpha(\theta_j)^{i_j} \cdot (1 - \tau_\alpha(\theta_j)^{2-i_j}$
      if $i_j = 1$ **then**
        $b_i := b_i \cdot \cos\alpha$
  return $C \cdot \boldsymbol{b} / \sum_i b_i$

must hold that

$$\omega_{2,(0)} = \omega_{2,(2)} \quad \text{and} \quad \omega_{2,(1)} = \cos\alpha \cdot \omega_{2,(2)},$$

for the chosen opening angle α. Reversely, the control points $\bar{b}_i(\theta_1)$ have to represent circular arcs as $b_{\theta_1^\star}(\theta_2)$ is rotated around the first axis for different values of $\theta_1^\star$. For the weights $\omega_{1,(i,j)}$ this means that

$$w_{1,(i,0)} = w_{1,(i,1)} \quad \text{and} \quad w_{1,(i,1)} = \cos\alpha \cdot w_{1,(i,1)}.$$

For the actual weights $\gamma_{(i,j)}$ this means that they can be defined as

$$\begin{aligned}
\gamma_{(0,0)} &= 1, & \gamma_{(0,1)} &= \cos\alpha, & \gamma_{(0,2)} &= 1,\\
\gamma_{(1,0)} &= \cos\alpha, & \gamma_{(1,1)} &= \cos^2\alpha, & \gamma_{(1,2)} &= \cos\alpha,\\
\gamma_{(2,0)} &= 1, & \gamma_{(2,1)} &= \cos\alpha, & \gamma_{(2,2)} &= 1,
\end{aligned}$$

or, more generally, as

$$\gamma_{\boldsymbol{i}} = \left(\cos\alpha\right)^{\text{count}(\boldsymbol{i})}, \tag{3.21}$$

where $\text{count}(\cdot)$ denotes a functions that returns the number of occurrences of the value one in an index vector. This concludes the definition of the rational tensor-product Bézier form developed in this thesis which will be called *Kinematic Bézier Maps* in this work from now on. Despite the complex mathematics described, the model and its learning algorithms are very minimalistic approaches that can be implemented and verified very easily as shown in Algorithm. 5 which describes the evaluation of a model in pseudo code.

Knowing the projective weights in advance saves the third step reducing the procedure to only linear operations which also opens the opportunity to find the least squares minimizers.
The fact that the projective weights are assigned to fixed values does not imply that every KBM model rep-

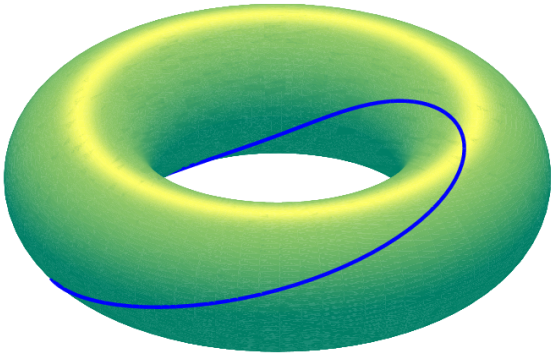


Figure 3.8.: Quartic curve on the surface of the torus generated from a linear trajectory in the configuration space.

resents a valid kinematics. In order to be a combination of circles, the vertices of the control net need to be aligned in a special way (see Section 3.1.3). This cannot always be enforced during learning and noisy data may lead to degenerated models. The projective weights, however, do enforce that every KBM form is a combination of ellipses.

Partial derivatives and Jacobian

The application of the Bézier form comes along with a set of useful theoretic tools for the manipulation of the control net. This comes in handy for the creation of the partial derivatives of the KBM model which can be represented by KBM models for themselves. The partial derivatives in the Jacobian matrix of a KBM model $b(\boldsymbol{\theta})$

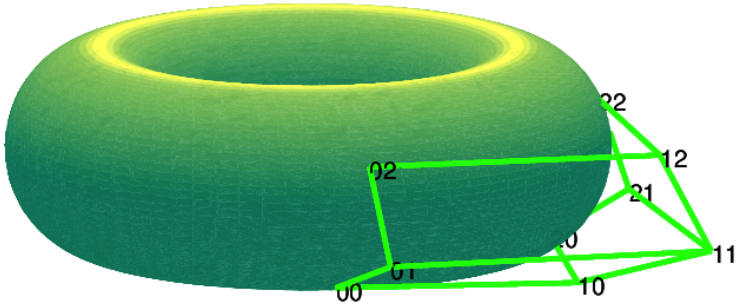
$$J_b(\boldsymbol{\theta}) := \left(\frac{\partial}{\partial \theta_1} b(\boldsymbol{\theta}), \ldots, \frac{\partial}{\partial \theta_{d_{\text{in}}}} b(\boldsymbol{\theta}) \right).$$

play an important role in the iterative numerical solution of the inverse kinematics. As it has to be frequently constructed for many different joint configurations, a representation has to be found that can be as quickly evaluated as the forward kinematics itself.

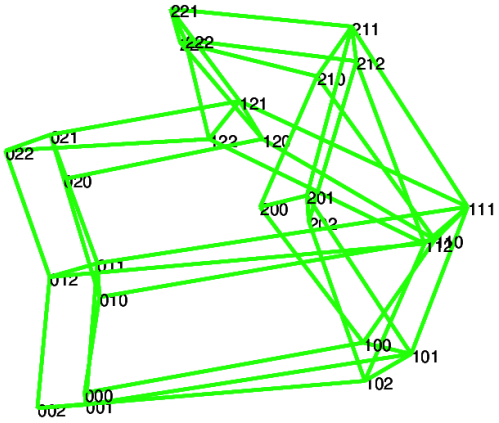
Affine derivative The partial derivatives of an affine quadratic Bézier tensor product is obtained by the construction of new control vertices by computing the forward differences between neighboring control vertices with respect to the respective joint angle θ_j of the partial derivative

$$\Delta_j \boldsymbol{b}_{(i_1,\ldots,i_{j-1},0,i_{j+1},\ldots,i_{d_{\text{in}}})} = \boldsymbol{b}_{(i_1,\ldots,i_{j-1},1,i_{j+1},\ldots,i_{d_{\text{in}}})} - \boldsymbol{b}_{(i_1,\ldots,i_{j-1},0,i_{j+1},\ldots,i_{d_{\text{in}}})} \tag{3.22}$$

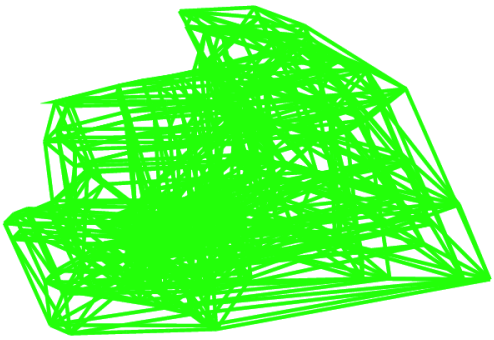
$$\Delta_j \boldsymbol{b}_{(i_1,\ldots,i_{j-1},1,i_{j+1},\ldots,i_{d_{\text{in}}})} = \boldsymbol{b}_{(i_1,\ldots,i_{j-1},2,i_{j+1},\ldots,i_{d_{\text{in}}})} - \boldsymbol{b}_{(i_1,\ldots,i_{j-1},1,i_{j+1},\ldots,i_{d_{\text{in}}})} \tag{3.23}$$



(a) Control net and manifold for two degrees of freedom.



(b) Control net for three degrees of freedom.



(c) Control net for five degrees of freedom (without labels).

Figure 3.9.: Control nets of KBM models for kinematics with randomly created transformations between the axis. The control nets for different numbers of degrees of freedom are shown. The manifold of the workspace can only be plotted for two degrees of freedom.

resulting in a rational Bézier form with $2 \cdot 3^{d_{\text{in}}-1}$ control vertices

$$\mathscr{C}_{\Delta,j} = \{\Delta_j \boldsymbol{b_i} : \boldsymbol{i} \in \mathscr{I}^{d_{\text{in}}}, i_j \leq 1\}$$
$$\Delta_j \boldsymbol{b_i} = \boldsymbol{b}_{\boldsymbol{i}+\boldsymbol{e}_j} - \boldsymbol{b_i}, \quad \text{with} \quad \boldsymbol{e}_j = \Big(0,\ldots,0,\underset{j}{1},0,\ldots,0\Big)$$

where $\boldsymbol{e}_j$ is the j-th unity vector in $\mathscr{I}^{d_{\text{in}}}$ used for greater compactness. This function is then linear with respect to the j-th joint. In order to be able to use the same basis functions as for the regular evaluation, a technique called *degree elevation* (Prautzsch et al., 2002) must therefore be applied to this control net first. This artificially raises the polynomial degree in this direction and has no influence on the models output—due to a property of the Bézier representation called *linear precision*. The definition of the set of control vertices and forward differences is modified to

$$\bar{\mathscr{C}}_{\Delta,j} = \{\overline{\Delta_j \boldsymbol{b_i}} : \boldsymbol{i} \in \mathscr{I}^{d_{\text{in}}},\}$$
$$\overline{\Delta_j \boldsymbol{b_i}} = \begin{cases} \boldsymbol{b}_{\boldsymbol{i}+\boldsymbol{e}_j} - \boldsymbol{b_i}, & i_j = 0, \\ 1/2(\boldsymbol{b}_{\boldsymbol{i}+\boldsymbol{e}_j} - \boldsymbol{b}_{\boldsymbol{i}-\boldsymbol{e}_j}), & i_j = 0, \\ \boldsymbol{b_i} - \boldsymbol{b}_{\boldsymbol{i}-\boldsymbol{e}_j}, & i_j = 2. \end{cases}$$

Rational derivative The partial derivative of rational Bézier tensor product—and the KBM model in particular—is decidedly more complex compared to the affine case as its derivatives are not linear functions:

$$\begin{aligned}\frac{\partial}{\partial\theta_j}b(\boldsymbol{\theta}) &= \frac{\partial}{\partial\theta_j}\frac{\sum_{\boldsymbol{i}\in\mathscr{I}_2^{d_\text{in}}}\boldsymbol{b_i}\cdot\gamma_{\boldsymbol{i}}\cdot(B_{\boldsymbol{i}}\circ\tau_\alpha)(\boldsymbol{\theta})}{\sum_{\boldsymbol{i}\in\mathscr{I}_2^{d_\text{in}}}\gamma_{\boldsymbol{i}}\cdot(B_{\boldsymbol{i}}\circ\tau_\alpha)(\boldsymbol{\theta})} =: \frac{\partial}{\partial\theta_j}\frac{a(\tau_\alpha(\boldsymbol{\theta}))}{b(\tau_\alpha(\boldsymbol{\theta}))}\\ &= \frac{\partial/\partial\theta_j\tau_\alpha(\theta_j)\cdot\left(\partial/\partial\theta_j a(\tau_\alpha(\boldsymbol{\theta}))\cdot b(\tau_\alpha(\boldsymbol{\theta})) - \partial/\partial\theta_j b(\tau_\alpha(\boldsymbol{\theta}))\cdot a(\tau_\alpha(\boldsymbol{\theta}))\right)}{b^2(\tau_\alpha(\boldsymbol{\theta}))}\\ \partial/\partial\theta_j\tau_\alpha(\theta_j) &= \frac{1}{\tan\alpha/2\cdot\cos^2\theta_j/2},\end{aligned}$$

where $a(\cdot)$ and $b(\cdot)$ are affine Bézier tensor product helper functions that can be derived as described above. Although the formula looks complex, its components can be computed quickly and some of them can be reused for all partial derivatives and evaluation (namely $a(\tau_\alpha(\boldsymbol{\theta}))$ and $b(\tau_\alpha(\boldsymbol{\theta}))$ which are already created when evaluating the KBM model at $\boldsymbol{\theta}$). This leads to an efficient way of obtaining the partial derivatives and the Jacobian matrix that can be vectorized and therefore be computed very quickly on modern processors.

Simplified rational derivative The partial derivatives can also be constructed more comfortably by further exploiting the a priori knowledge. Assuming, without loss of generality, that a univariate KBM $b(\theta)$ perfectly represents a unit circle

$$b(\theta) \equiv \Big(\cos\theta, \sin\theta\Big)^T,$$

then

$$\frac{\partial}{\partial\theta}b(\theta) = \Big(-\sin\theta, \cos\theta\Big)^T$$

holds for every θ. Figure 3.2c shows how the derivative can be constructed from the forward differences. The intermediate control point has to be chosen such that the circular condition is met and the whole control polygon has to be scaled. Applied to the multivariate case, this leads to the following definition of the control net

$$\begin{aligned}\tilde{\mathscr{C}}_{\Delta,j} &= \{\tilde{\Delta}_j\boldsymbol{b_i} : \boldsymbol{i}\in\mathscr{I}^{d_\text{in}}\}\\ \tilde{\Delta}_j\boldsymbol{b_i} &= \frac{1}{\tan\alpha}\cdot\begin{cases}\boldsymbol{b}_{\boldsymbol{i}+\boldsymbol{e}_j}-\boldsymbol{b_i}, & i_j=0,\\ \frac{1}{2\cdot\cos^2\alpha}(\boldsymbol{b}_{\boldsymbol{i}+\boldsymbol{e}_j}-\boldsymbol{b}_{\boldsymbol{i}-\boldsymbol{e}_j}), & i_j=0,\\ \boldsymbol{b_i}-\boldsymbol{b}_{\boldsymbol{i}-\boldsymbol{e}_j}, & i_j=2.\end{cases}\end{aligned}$$

The KBM using this control vertices represents the partial derivative with respect to the j-th joint and no additional calculus is required. However, this only holds if and only if the KBM perfectly represents a kinematics and is only an approximation otherwise.

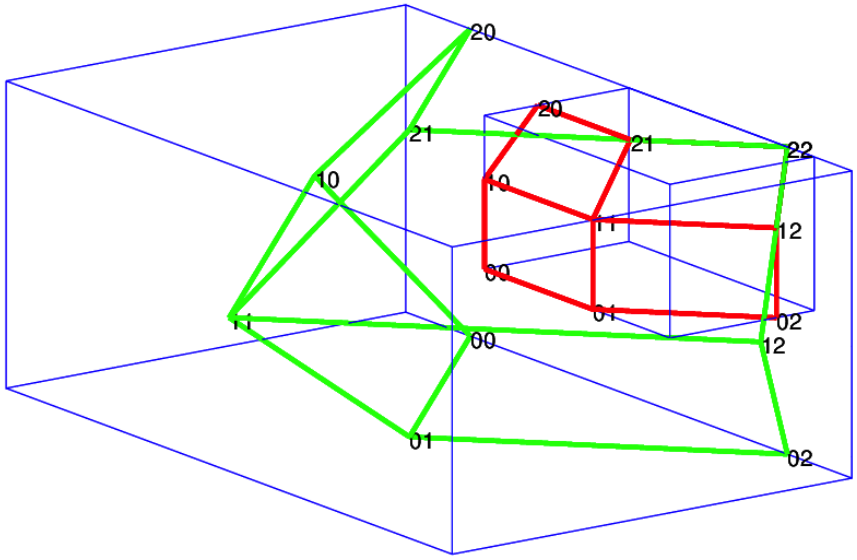


Figure 3.10.: The subdivision operation applied to the KBM model of an exemplary robot with two orthogonal joint axis. The original control net *(green)* is subdivided to represent a smaller region of the configuration space *(red)*. All outputs generated by the two models have to lie in the convex hull and their bounding boxes.

Subdivision

So far, it has been demonstrated how the control net can be modified by degree elevation and computing forward differences for the derivatives. Another useful algorithm that can be applied to the net is called *'subdivision'*. This operation subdivides the support Bézier curve (or manifold) into smaller segments together with the respective control nets. The example for two degrees of freedom in Figure 3.10 gives and impression of this operation. In addition to the original control net (green) and the subdivided net (red), the axes aligned bounding boxes (blue) are shown in this image. Note that both control nets represent the same kinematics. The subdivision can be computed with a variant of DeCasteljau's algorithm and a modification of the polynomial basis.

The procedure will be shown for the univariate case but can be transferred analogously to multivariate KBM models. Without loss of generality, a KBM is defined over the support $[-\alpha,+\alpha]$. All images of the angles out of this interval lie in the convex hull of the control points (convexity property). A subdivision is defined over new boundaries of the support interval $[\beta_l,\beta_u]$ where

$$[\beta_l,\beta_u] \subset [-\alpha,\alpha].$$

In order to determine the control points of the resulting segment, the formalism of the KBM is modified:

$$\bar{b}(\theta_1,\theta_2) = \frac{\sum_i \gamma_i \cdot \boldsymbol{b}_i \cdot (\bar{B}_i \circ \tau_\alpha)(\theta_1,\theta_2)}{\sum_i \gamma_i \cdot (\bar{B}_i \circ \tau_\alpha)(\theta_1,\theta_2)}$$

where

$$(\bar{B}_i \circ \tau_\alpha)(\theta_1,\theta_2) := \binom{2}{i} \cdot \big(\tau_\alpha(\theta_1)\big)^i \cdot \big(1-\tau_\alpha(\theta_2)\big)^{2-i}.$$

The new function $\bar{b}(\cdot,\cdot)$ is a symmetric rational polynomial, that is

$$\bar{b}(\theta_1,\theta_2) = \bar{b}(\theta_2,\theta_1),$$

that equals the original form when the three arguments are equal

$$b(\theta) = \bar{b}(\theta,\theta).$$

This form that depends on twice as many input variables is called the *blossom* (Prautzsch et al., 2002) of the KBM. The control points of $b(\theta)$ can be expressed by the blossom

$$\boldsymbol{b}_0 = \bar{b}(-\alpha,-\alpha), \qquad \boldsymbol{b}_1 = \bar{b}(-\alpha,+\alpha), \qquad \boldsymbol{b}_2 = \bar{b}(+\alpha,+\alpha)$$

Now the segment $s(\cdot)$ over $[\beta_l,\beta_u]$ can be constructed by new control points and an offset in the joint angles

$$s(\theta) = \frac{\sum_i \tilde{\gamma}_i \cdot \boldsymbol{s}_i \cdot (B \circ \tau_{\tilde{\alpha}}) \left(\theta - \frac{\beta_l+\beta_2}{2}\right)}{\sum_i \tilde{\gamma}_i (B \circ \tau_{\tilde{\alpha}}) \left(\theta - \frac{\beta_l+\beta_2}{2}\right)}$$

where

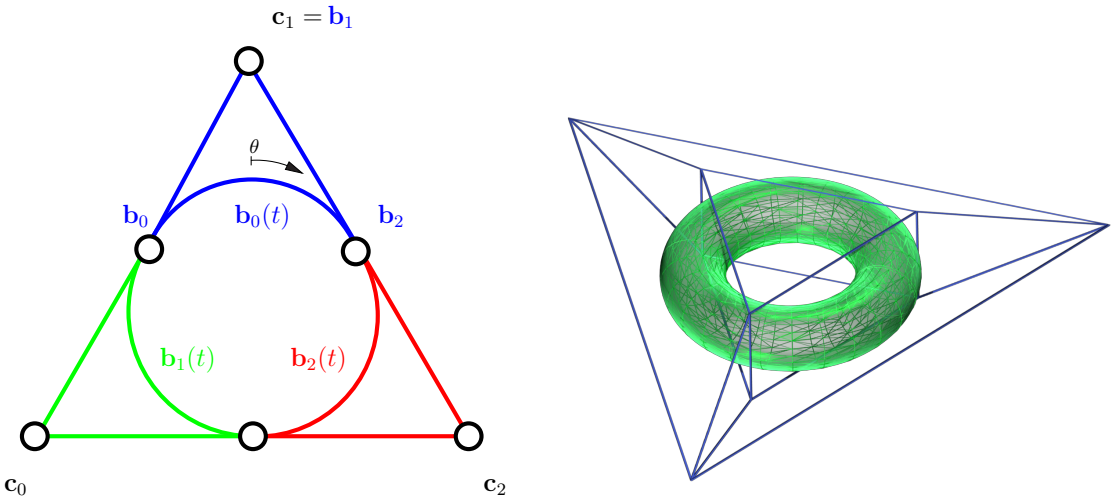
$$\boldsymbol{s}_0 = \bar{b}(\beta_l,\beta_l), \qquad \boldsymbol{s}_1 = \bar{b}(\beta_l,\beta_u), \qquad \boldsymbol{s}_2 = \bar{b}(\beta_u,\beta_u),$$
$$\tilde{\alpha} = \frac{\beta_u - \beta_l}{2} \Longrightarrow \tilde{\gamma}_1 = \cos(\tilde{\alpha}).$$

Note that the blossom can also be used to extrude the KBM model, that is, its definition over a bigger support interval.

The subdivision is very useful because of the convex hull property of the KBM representation: All possible function values in the support interval lie in the convex hull of the control net. If a function value lies outside the convex hull, it cannot result from inputs within the support interval. By successively decreasing the support interval, this can be exploited to find all (quantized) solutions to the inverse problem in an interval analysis. Instead of the convex hull, the bounding box (see Figure 3.10) or an enlarged bounding box (to account for sensor noise) can be used as an approximation. A bounding box can be computed much faster, but it has to be constructed from the 3^n control points such that this method becomes slow for many degrees of freedom. Unlike interval arithmetics (Merlet, 2009), however, the volume convex hull strictly decreases when subdividing.

Bézier-spline form

The definition of the KBM in Eq. (3.19) features a single user-defined hyper parameter α—the opening angle of the control net. Ideally, this angle should be chosen differently for each degree of freedom and be picked according to the distribution of the training data to propel the iterative numeric learning rule in Sec-



(a) Schematic of the spline form of the KBM for a single degree of freedom.

(b) Control net and workspace manifold for two degrees of freedom.

Figure 3.11.: The subdivision of a circle into the three segments of the *Bézier spline* representation. *Source: (Ulbrich et al., 2012c)*

tion 4.3. From this hyper parameter arises a problem with the numerical stability when single joint angles are close to $\pm\pi$ as the values of the parameter transformation $\tau_\alpha(\cdot)$ (Eq. (3.15)) reach too large values. In addition, a model that is free of user-defined hyper parameters for tuning is more convenient to handle its application.

For these reasons, an alternate variant of the KBM model, which is free of hyper parameters, will be proposed in this paragraph. It is based on *Bézier splines* (not to be confused with *b-splines*). For each degree of freedom, the model consists of the connection of several—in this case, three—Bézier curves[2] that all represent the same circle and whose convex hulls formed by the control polygons cover the whole circle (see Figure 3.11a). This leads to the definition of new rational basis functions but otherwise leaves the model and its properties as shown before intact including modeling the inverse dynamics (Section 3.2.2) and learning algorithms that do not explicitly exploit the geometry of the control net (Section 4.1).

Univariate form As mentioned before, the key to this form is the subdivision of a circle into three congruent segments which are approximated by a univariate KBM model each such that the convex hulls of the control points are congruent, disjoint and cover the whole circle. This restricts the hyper parameters of the models to $\pi/3$. The three segments are defined by the open joint angle intervals

$$[-\pi, -\pi/3), \quad [-\pi/3, \pi/3), \quad \text{and} \quad [\tfrac{\pi}{3}, \pi)$$

and for each a model with a linear parameter transformation is defined by

$$b_1(\theta + 2/3\pi), \quad b_2(\theta), \quad \text{and} \quad b_3(\theta + 2/3\pi).$$

[2]Note that this is the minimal number of spline segments as the wideness parameter α is below 90° and all control points have affine images.

The resulting configuration has six control points as shown in Figure 3.11a. Because of the properties that the end points of a Bézier polygon lie on the curve and that the connection between the control points are tangent vectors, three points are collinear. This can be exploited to reduce the number of control points back to three independent control points $\boldsymbol{c}_i$ which otherwise would have led to an increase in the demand of training samples. The first model is then of the form

$$b_1(\theta + 2/3\pi) = \frac{\boldsymbol{c}_0 \cdot B_0(\tau_{2/3\pi}(\theta + 2/3\pi)) + \cos\alpha \cdot \boldsymbol{c}_1 \cdot B_1(\tau_{2/3\pi}(\theta + 2/3\pi)) + \boldsymbol{c}_2 \cdot B_2(\tau_{2/3\pi}(\theta + 2/3\pi))}{B_0(\tau_{2/3\pi}(\theta + 2/3\pi)) + \cos\alpha \cdot B_1(\tau_{2/3\pi}(\theta + 2/3\pi)) + \cdot B_2(\tau_{2/3\pi}(\theta + 2/3\pi))}, \quad (3.24)$$

where, after applying the constraints,

$$\boldsymbol{b}_0 = \frac{\boldsymbol{c}_0 + \boldsymbol{c}_1}{2}, \quad \boldsymbol{b}_1 = \boldsymbol{c}_1 \quad \text{and} \quad \boldsymbol{b}_2 = \frac{\boldsymbol{c}_2 + \boldsymbol{c}_1}{2}.$$

Now the polynomial basis is adjusted such that it reflects the new control points $\boldsymbol{c}_i$:

$$\begin{aligned}
C_0^0(\theta) &:= \tfrac{1}{2} \cdot B_0(\tau_{2/3\pi}(\theta + 2/3\pi)), \\
C_1^0(\theta) &:= \tfrac{1}{2} \cdot B_0(\tau_{2/3\pi}(\theta + 2/3\pi)) + \cos\alpha \cdot B_1(\tau_{2/3\pi}(\theta + 2/3\pi)) + \tfrac{1}{2} \cdot B_2(\tau_{2/3\pi}(\theta + 2/3\pi)) \\
&= \tfrac{1}{2} \cdot \tau_{2/3\pi}(\theta)^2 + \cos\pi/3 \cdot 2(1 - \tau_{2/3\pi}(\theta + 2/3\pi)) \cdot \tau_{2/3\pi}(\theta + 2/3\pi) \\
&\quad + \tfrac{1}{2}(1 - \tau_{2/3\pi}(\theta + 2/3\pi))^2 \\
&= \tfrac{1}{2}(\tau_{2/3\pi}(\theta + 2/3\pi) + (1 - \tau_{2/3\pi}(\theta + 2/3\pi)))^2 \\
&= \tfrac{1}{2}, \\
C_2^0(\theta) &:= \tfrac{1}{2} \cdot B_2(\tau_{2/3\pi}(\theta + 2/3\pi)),
\end{aligned} \quad (3.25)$$

which leads to final equation of this spline segment

$$b_1(\theta + 2/3\pi) = \frac{\boldsymbol{c}_0 \cdot C_0^0(\theta) + \boldsymbol{c}_1 \cdot C_0^0(\theta) + \boldsymbol{c}_2 \cdot C_2^0(\theta)}{C_0^0(\theta) + C_0^0(\theta) + C_2^0(\theta)}$$

The other spline segments $b_2(\cdot)$ and $b_3(\cdot)$ and basis $C_i^1(\cdot)$ and $C_i^2(\cdot)$ are analogously defined.
Now, the complete spline can be defined as

$$c(\theta) = \frac{\boldsymbol{c}_0 \cdot C_0(\theta) + \boldsymbol{c}_1 \cdot C_1(\theta) + \boldsymbol{c}_2 \cdot C_2^0(\theta)}{C_0(\theta) + C_1(\theta) + C_2^0(\theta},$$

where the C_i are the new rational basis functions already combined with the parameter transformation $\tau_{2/3\pi}(\cdot)$ which are defined for each segment individually as

$$C_0(\theta) = \begin{cases} 1/2, & \theta \in [-\pi, -\frac{\pi}{3}) \\ 1/2 \cdot B_0(\tau_{2/3\pi}(\theta)), & \theta \in [-\frac{\pi}{3}, \frac{\pi}{3}) \\ 1/2 \cdot B_2(\tau_{2/3\pi}(\theta - \frac{\pi}{3})), & \theta \in [\frac{\pi}{3}, \pi) \end{cases}$$

$$C_1(\theta) = \begin{cases} 1/2 \cdot B_2(\tau_{2/3\pi}(\theta + \frac{\pi}{3})), & \theta \in [-\pi, -\frac{\pi}{3}) \\ (\tau_{2/3\pi}(\theta))1/2, & \theta \in [-\frac{\pi}{3}, \frac{\pi}{3}) \\ 1/2 \cdot B_0(\tau_{2/3\pi}(\theta - \frac{\pi}{3})), & \theta \in [\frac{\pi}{3}, \pi) \end{cases}$$

$$C_2(\theta) = \begin{cases} 1/2 \cdot B_0(\tau_{2/3\pi}(\theta + \frac{\pi}{3})), & \theta \in [-\pi, -\frac{\pi}{3}) \\ 1/2 \cdot B_2(\tau_{2/3\pi}(\theta)), & \theta \in [-\frac{\pi}{3}, \frac{\pi}{3}) \\ 1/2, & \theta \in [\frac{\pi}{3}, \pi) \end{cases}$$

Multivariate form All techniques presented considering the KBM model so far still apply for the spline form of the KBM by a simple replacement of the rational basis and former weights. This includes the tensor product representation with the formula

$$b(\boldsymbol{\theta}) = \frac{\sum_{\boldsymbol{i} \in \mathscr{I}_2^{d_{\text{in}}}} \boldsymbol{c_i} \cdot C_{\boldsymbol{i}}(\boldsymbol{\theta})}{\sum_{\boldsymbol{i} \in \mathscr{I}_2^{d_{\text{in}}}} C_{\boldsymbol{i}}(\boldsymbol{\theta})} \tag{3.26}$$

and the derivatives. An image of a multivariate workspace of the exemplary robot with two orthogonal axes is shown in Figure 3.11b.

Representation of orientation

So far, both the control vertices and the return value of the KBM models were assumed to be spatial points with output dimension d_{out}. However its is possible and very often desired to extend this representation to cover the complete spatial pose, that is, to also include the orientation of the end-effector. This paragraph covers three different methods for the representation of the orientation: *homogenous transformation matrices*, *Euler angles* and *quaternions*.

Homogenous transformation matrices The most simple way to achieve this is the transformation of the end-effector poses (which of course have to be observable by the tracking system) into homogeneous transformation matrices such that the set of training data is of the form

$$\mathscr{T} = \left\{ (\boldsymbol{\theta}_i, X_i) : \boldsymbol{\theta}_i \in \Theta^{d_{\text{in}}}, X_i \in \mathbb{R}^{4\times 4}, i \in \{1, \ldots, m\} \right\}. \tag{3.27}$$

Switching to matrices rather than vectors has no impact on the KBM models which, in analogy to Eq. (3.19), becomes

$$\begin{aligned} b(\cdot): \quad & \Theta^{d_{\text{in}}} \to \mathbb{R}^{4\times 4}, \\ \boldsymbol{\theta} \mapsto & \frac{\sum_{\boldsymbol{i}\in\mathscr{I}_2^{d_{\text{in}}}} T_{\boldsymbol{i}} \cdot \gamma_{\boldsymbol{i}} \cdot (B_{\boldsymbol{i}} \circ \tau_\alpha)(\boldsymbol{\theta})}{\sum_{\boldsymbol{i}\in\mathscr{I}_2^{d_{\text{in}}}} \gamma_{\boldsymbol{i}} \cdot (B_{\boldsymbol{i}} \circ \tau_\alpha)(\boldsymbol{\theta})}, \\ \mathscr{C} = & \left\{ T_{\boldsymbol{i}} : \boldsymbol{i} \in I^{d_{\text{in}}} \right\}, \end{aligned}$$

which indicates that the control vertices in the parameter set $\mathscr{C}$ are now homogeneous matrices. From this representation, arise several problems though. First, the representation with homogenous transformation matrices itself is redundant. The can admittedly be reduced to the number of 12 by discarding the last row but six coordinates completely suffice to describe a spatial pose. Second, depending on how well the KBM approximates its target function, the return value $b(\boldsymbol{\theta})$ is not necessarily a valid transformation matrix in general (e.g., when learning from very noisy data). An orthonormalization (by a Gram-Schmidt process or singular value decomposition for instance) is then highly advisable.

Euler angles Euler angles are a more efficient representation of spatial poses compared to homogenous matrices. The orientation hereby is described by three rotation angles α, β and γ around predefined axis. There are many conventions on how these axis are chosen. For instance,

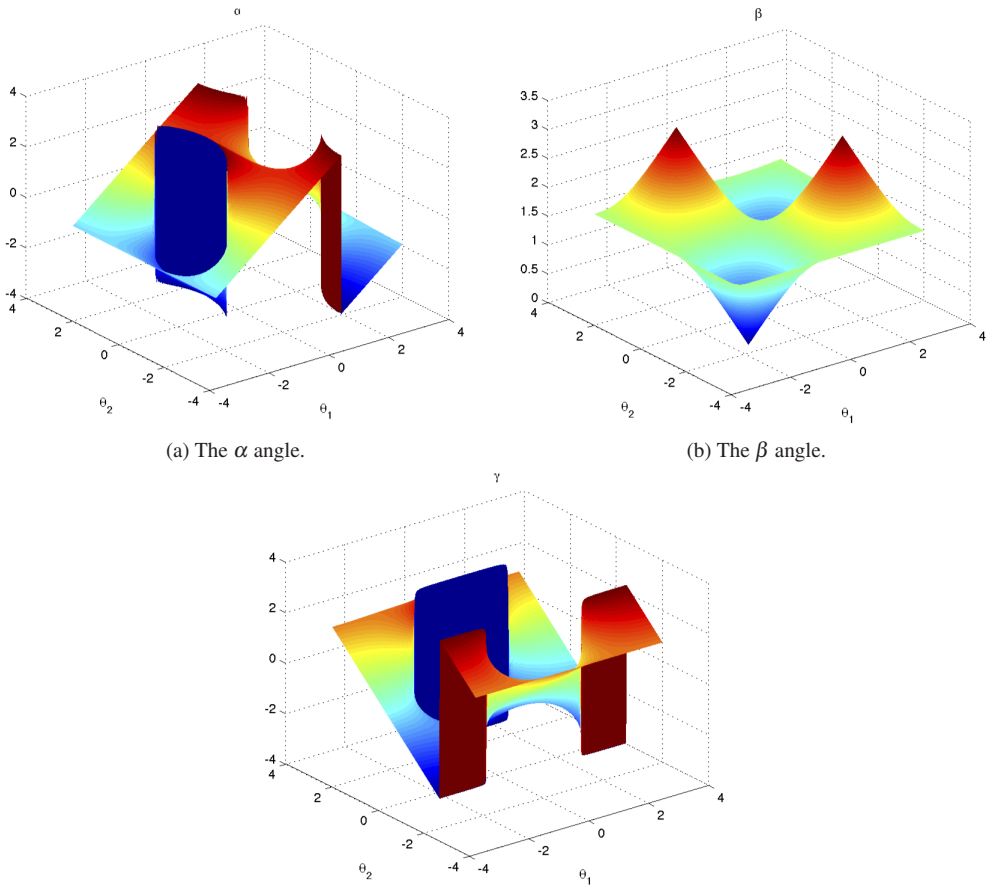
$$R_x(\alpha) \cdot R_{z'}(\beta) \cdot R_{x''}(\gamma), \tag{3.28}$$

which denotes a rotation around the global x-axis followed by a rotation around the new local z-axis z' and rotation by the then resulting new local x-axis x''. Closely related is the *Roll-Pitch-Yaw* convention with rotates around the global coordinate system's axes. The Euler angles are not unambiguous. In Eq. (3.28) for instance, there are infinitely many pairs of α and γ representing the same orientation if β equals zero. This is the so called *gimbal lock*. More severe, however, is that the manifold of the Euler angles over the generating joint angles is not continuous (see Figure 3.12) which immediately means that they cannot be represented by the KBM models.

Quaternions Another compact representation for orientations are *quaternions*. They represent a orientation with four coordinates which makes them slightly more redundant as the Euler angles but, unlike them, the representation is unique with exception of

$$q \equiv -q$$

which can be resolved by choosing only the quaternions of the upper hemisphere of the unit sphere. Their exact definition and those of the operations that can be applied to them is out of the scope of this work. But, again unlike the Euler angles, their components form continuous manifolds over the generating joint angles which is depicted in Figure 3.13 for two degrees of freedom. Because of this property, the quaternion representation can exactly be expressed by the KBM model without modification.



(a) The α angle.

(b) The β angle.

(c) The γ angle.

Figure 3.12.: Exemplary manifolds spanned by each Euler angle over the two joint angles of a kinematics with two degrees of freedom using the z, y', z''-convention. The manifolds are not continuous and therefore, cannot be expressed by a KBM.

Reconstruction of the orientation For learning the orientation of the end-effector, it is of course necessary that it can be observed by the robot's tracking system. However, even if this is not the case, the orientation can be reconstructed from an existing three-dimensional KBM model. Therefore, a coordinate system is created that directly lies in the output of the model and that is represented by a homogenous transformation matrix. The method that reconstructs the orientation requires two non-parallel vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ that depend on all joint angles of the kinematics and are rotated around all joint axis.The first vector $\boldsymbol{v}_1$ given a joint configuration $\boldsymbol{\theta}$ can be expressed as a function describing the partial derivatives of the model

$$v_1(\boldsymbol{\theta}) = \frac{\partial}{\partial \theta_{d_{\text{in}}}} b(\boldsymbol{\theta})$$

where $\theta_{d_{\text{in}}}$ has to be the last rotation axis, that is, the joint angles have to be sorted by their lateral distance—the number of joints separating them from the origin of the kinematics, see Chapter 5 and Eq. (3.29)). The second angle $\boldsymbol{v}_2$ must lie in the plane spanned by all $v_1(\boldsymbol{\theta})$ when all joint angles but the last are fixed. Eq. (3.20) shows how a tensor product can be formed into a univariate curve for $d_{\text{in}} - 1$ fixed joint positions for two degrees of freedom. The vector v_2 can be then, for instance, obtained by the difference between two control vertices. Another, easier possibly is to add an offset $\theta_\star$ to $\theta_{d_{\text{in}}}$ and assign the partial derivative to $\boldsymbol{v}_2$.

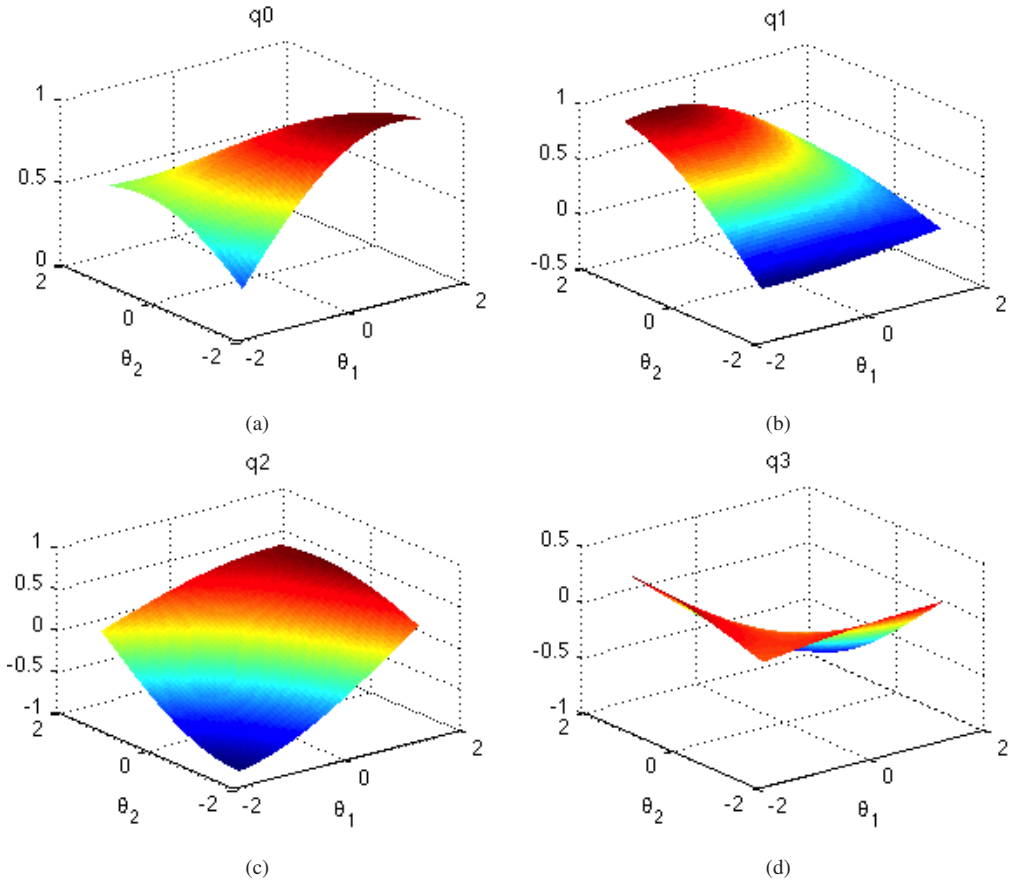


Figure 3.13.: The manifold of the quaternion components over two joint angles. The quaternions generate a continuous manifold and are suitable for learning with the KBM model.

In dependence of the current joint configuration $\boldsymbol{\theta}$, $\boldsymbol{v}_2$ can be expressed by the function

$$v_2(\boldsymbol{\theta}) = \frac{\partial}{\partial \theta_{d_{\text{in}}}} b(\bar{\boldsymbol{\theta}}), \quad \text{where} \quad \bar{\boldsymbol{\theta}} = \left(\theta_1, \ldots, \theta_{d_{\text{in}}-1}, \theta_{d_{\text{in}}} + \theta_\star\right).$$

Once $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are determined the coordinate system can be constructed as

$$\begin{aligned}
\boldsymbol{e}_1(\boldsymbol{\theta}) &= \frac{v_1(\boldsymbol{\theta})}{\|v_1(\boldsymbol{\theta})\|}, \\
\boldsymbol{e}_3(\boldsymbol{\theta}) &= \frac{v_1(\boldsymbol{\theta}) \times v_2(\boldsymbol{\theta})}{\|v_1(\boldsymbol{\theta}) \times v_2(\boldsymbol{\theta})\|}, \\
\boldsymbol{e}_2(\boldsymbol{\theta}) &= \boldsymbol{e}_1(\boldsymbol{\theta}) \times \boldsymbol{e}_3(\boldsymbol{\theta}), \\
T(\boldsymbol{\theta}) &= \begin{pmatrix} \boldsymbol{e}_1(\boldsymbol{\theta}) & \boldsymbol{e}_1(\boldsymbol{\theta}) & \boldsymbol{e}_1(\boldsymbol{\theta}) & b(\boldsymbol{\theta}) \\ 0 & 0 & 0 & 1 \end{pmatrix}.
\end{aligned}$$

This resulting matrix T depends on the joint configuration $\boldsymbol{\theta}$ and represents an orthogonal coordinate system associated to the end-effector such that there exists a constant transformation matrix C for which holds

$$T(\boldsymbol{\theta}) \cdot C = k(\boldsymbol{\theta}),$$

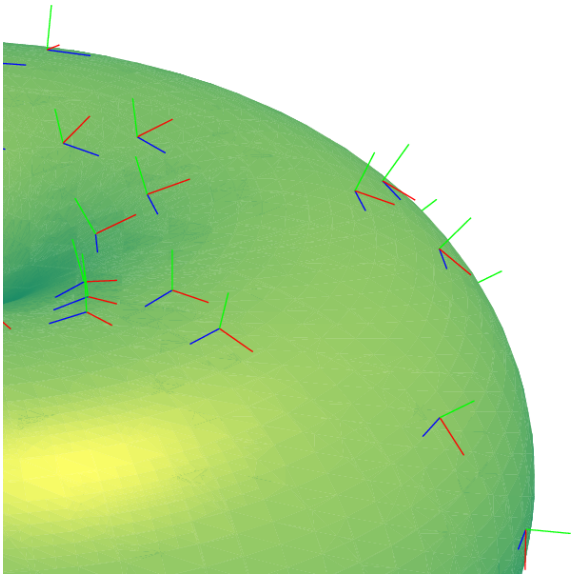


Figure 3.14.: Reconstructed frames of the surface of a learned manifold for a kinematics with two orthogonal joints.

where $k(\boldsymbol{\theta})$ denotes the latent coordinate system of the end-effector

$$k(\boldsymbol{\theta}) = M_{0,1}(\theta_1) \cdot \ldots \cdot M_{d_{\text{in}}-1,d_{\text{in}}}(\theta_{d_{\text{in}}}), \quad M_{\square} \in SE_3. \tag{3.29}$$

3.2.2. Modeling the inverse dynamics

For safe and efficient control of robots and planning, an accurate model of the robot's dynamic behavior is as crucial as knowing forward kinematics. Compared to the forward kinematics, the dynamics has additional great number of model parameters that are difficult to determine such as the inertial tensor, center of mass and the weight of each arm element, which change more easily if additional objects are attached to the arm. As a consequence, this makes the dynamic a useful target for machine learning.
The KBM presented so far can be only applied to kinematics but the dynamic properties depend at any time on the kinematic configuration of the robot. In this section, it will be shown how the principles of the KBM can be transferred to inverse robot dynamics starting with an introduction on the Lagrangian equations of motion.

Robot Dynamics

The inverse dynamics of a robot arm describes the relation between a robot's joint configuration and motion encoded in its joint positions $\boldsymbol{\theta}$, and its first and second order derivatives, the joint velocity $\dot{\boldsymbol{\theta}}$ and acceleration $\ddot{\boldsymbol{\theta}}$ respectively. Subsumed in the tuple $\left(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}\right)$ they are called *generalized coordinates* and their

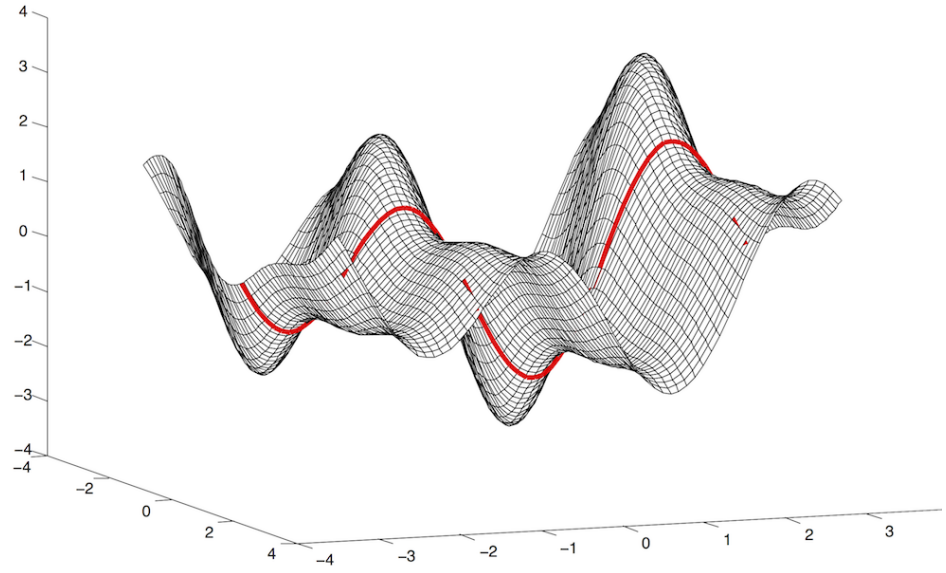


Figure 3.15.: Plot of the Coriolis component depending on two joint values when their derivatives equal zero.

relation to the *generalized forces* $\boldsymbol{q} = (q_1, \ldots, q_{d_{\text{in}}})$, that is, the torques and forces responsible for the current dynamic state, is expressed in the *Lagrangian equation* (Hollerbach, 1980)

$$Q_i \quad = \quad \frac{\mathrm{d}}{\mathrm{d}t}\left(\frac{\partial L(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}})}{\partial \dot{\theta}_i}\right) \quad - \quad \frac{\partial L(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}})}{\partial \theta_i}, \qquad \text{with } L(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}}) \quad = \quad T(\boldsymbol{\theta}) \quad - \quad V(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}})$$

where T and V are the kinetic and potential every in the system respectively which are both functions of the n generalized coordinates. This equation leads to the *Lagrangian equations of motion* which serve as the starting point for the creation of the dynamic model

$$\boldsymbol{q} \quad = \quad \underbrace{\boxed{M(\boldsymbol{\theta}) \cdot \ddot{\boldsymbol{\theta}}}}_{\text{Inertia } \boldsymbol{q}^M} + \underbrace{\boxed{C(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \cdot \dot{\boldsymbol{\theta}}}}_{\text{Coriolis } \boldsymbol{q}^C} + \underbrace{\boxed{\boldsymbol{h}(\boldsymbol{\theta})}}_{\text{Gravity } \boldsymbol{q}^g} + \underbrace{\boxed{\boldsymbol{\varepsilon}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}})}}_{\text{Friction}}, \quad M(\boldsymbol{\theta}), \quad C(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}) \quad \in \quad \mathbb{R}^{d_{\text{in}} \times d_{\text{in}}}, \tag{3.30}$$

where

$M(\boldsymbol{\theta})$ is a symmetrical and positive definite matrix that describes the robot's current inertial state,
$C(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}})$ denotes the centripetal and Coriolis forces coupling matrix, and
$\boldsymbol{h}(\boldsymbol{\theta})$ denotes d_{in}-dimensional vector that describes the gravity force acting on the system.

Additional nonlinear influences acting the of the dynamics such as backlash or friction, which are not all well understood, are subsumed in the error vector $\boldsymbol{\varepsilon}(\boldsymbol{\theta}, \dot{\boldsymbol{\theta}}, \ddot{\boldsymbol{\theta}})$ and are neglected for the creation sensorimotor model (i.e., have to be modeled separately). More detailed introduction on this topic is given in various literature, in Hollerbach (1980) for instance.

Learning the components of the Equations of Motion

In order to derive the extensions to the KBM necessary for learning the inverse dynamics, the equations of motions have to be analyzed in more detail. The following form, presented by (Hollerbach, 1980), yields how the equations are connected to the forward kinematics which will eventually lead to the transformation into a linear form based on the KBM that can be used for learning

$$q_i = \sum_{j=1}^{n} \Bigg(\underbrace{\boxed{\sum_{k=1}^{j} \left(\mathrm{tr}\left(\frac{\partial W_j}{\partial \theta_i} J_j \frac{\partial W_j}{\partial \theta_k}^T \right) \ddot{\theta}_k \right)}}_{\boldsymbol{q}^M} + \underbrace{\boxed{\sum_{k=1}^{j} \sum_{l=1}^{j} \left(\mathrm{tr}\left(\frac{\partial W_j}{\partial \theta_i} J_j \frac{\partial^2 W_j}{\partial \theta_k \partial \theta_l}^T \right) \dot{\theta}_k \dot{\theta}_l \right)}}_{\boldsymbol{q}^C} - \underbrace{\boxed{m_j \boldsymbol{g}^T \frac{\partial W_j}{\partial \theta_i} \boldsymbol{c}_j}}_{\boldsymbol{q}^g} \Bigg) \tag{3.31}$$

For each rigid body, the coordinate system W_j describing its pose with respect to the origin. Its center lies in the predecessing joint and is given as a a product of homogeneous matrices (e.g., following the Denavit-Hartenberg conventions) describing the coordinate transformations between succeeding links (partial forward kinematics)

$$W_j(\theta_1, \ldots, \theta_j) = M_{0,1}(\theta_1) \cdot M_{1,2}(\theta_2) \cdot \ldots \cdot M_{j-1,j}(\theta_j), \quad M_{\square} \in SO_3.$$

The following parameters are defined with respect to this coordinate frame

$J_j \in \mathbb{R}^{4\times4}$	the inertia tensor of the j-th link,
m_j	its mass and
$\boldsymbol{c}_j$	the (homogeneous) coordinates of the center of mass.

The force vector of the gravity is denoted with $\boldsymbol{g}$. This equation already reveals the close interconnection to the forward kinematics but it should be noted that it appears in its squared form in the inertia and Coriolis components $\boldsymbol{q}^M$ and $\boldsymbol{q}^C$. Starting from this equation, the dynamics will be first decomposed into its inertial, Coriolis and gravity terms $\boldsymbol{q}^M$, $\boldsymbol{q}^C$ and $\boldsymbol{q}^g$. For each term, it will be shown how they, for themselves, can be transformed into a learnable form linear in the generalized coordinates. Afterwards, the model for the complete inverse dynamics that learns all components simultaneously is assembled into a single model.

Gravity The gravity component $\boldsymbol{q}^g$ has the closest relation to the KBM model. It is one of the most dominant terms in the equations of motion and is also present if the robot is not in motion and all other terms are zero. Consequently it find an useful application, for instance, in learning gravity compensation and can be learned easy form truing samples collected at moments where the robot is not moving. From its definition

$$q_i^g = -\sum_{j=1}^{n} \left(m_j \boldsymbol{g}^T \frac{\partial \boxed{W_j}}{\partial \theta_i} \boldsymbol{c}_j \right) \tag{3.32}$$

it can be seen that it consists of the summed partial derivatives of the partial forward kinematics of the respective joint'c centers of mass multiplied with its weight m_j projected along the gravity vector $\boldsymbol{g}$. These

are all affine transformation which not interfere with the KBM modeling. The sum of multiple KBM models or their derivates is also a KBM due to its linearity

$$\begin{aligned}\boldsymbol{b}(\boldsymbol{\theta})+\boldsymbol{c}(\boldsymbol{\theta}) &= \frac{\sum_{\boldsymbol{i}\in\mathscr{I}^{d_{\text{in}}}} \gamma_{\boldsymbol{i}}\cdot\boldsymbol{b}_{\boldsymbol{i}}\cdot B_{\boldsymbol{i}}(\boldsymbol{\theta})}{\sum_{\boldsymbol{i}\in\mathscr{I}^{d_{\text{in}}}} \gamma_{\boldsymbol{i}}\cdot B_{\boldsymbol{i}}(\boldsymbol{\theta})} + \frac{\sum_{\boldsymbol{i}\in\mathscr{I}^{d_{\text{in}}}} \gamma_{\boldsymbol{i}}\cdot\boldsymbol{c}_{\boldsymbol{i}}\cdot B_{\boldsymbol{i}}(\boldsymbol{\theta})}{\sum_{\boldsymbol{i}\in\mathscr{I}^{d_{\text{in}}}} \gamma_{\boldsymbol{i}}\cdot B_{\boldsymbol{i}}(\boldsymbol{\theta})} \\ &= \frac{\sum_{\boldsymbol{i}\in\mathscr{I}^{d_{\text{in}}}} \gamma_{\boldsymbol{i}}\cdot(\boldsymbol{b}_{\boldsymbol{i}}+\boldsymbol{c}_{\boldsymbol{i}})\cdot B_{\boldsymbol{i}}(\boldsymbol{\theta})}{\sum_{\boldsymbol{i}\in\mathscr{I}^{d_{\text{in}}}} \gamma_{\boldsymbol{i}}\cdot B_{\boldsymbol{i}}(\boldsymbol{\theta})} \quad \square\end{aligned}$$

Thus, modeling the gravity component does not require any special modifications to the KBM form. As such, it can be learned with the same minimal amount of training data $3^{d_{\text{in}}}$. For the reason of distinction, the upper index of the basis functions $B^2_{\boldsymbol{i}}(\boldsymbol{\theta}_i)$ indicating their polynomial degree will not be omitted in the following paragraphs

The Intertia Matrix The inertial component $\boldsymbol{q}^M$ cannot be measured separately of $\boldsymbol{q}^g$ or $\boldsymbol{q}^C$ and consequently not be learned outside simulation. Its definition in Eq. (3.31)

$$q_i^M = \sum_{j=1}^{n}\left(\underbrace{\sum_{k=1}^{j}\operatorname{tr}\left(\frac{\partial \boxed{W_j}}{\partial\theta_i}J_j\frac{\partial \boxed{W_j^T}}{\partial\theta_k}\right)\boxed{\ddot{\theta}_k}}_{=:\mu_{i,j,k}}\right)$$

can be transformed in the following way

$$\begin{aligned}\boldsymbol{q}^M &= \begin{pmatrix} \mu_{1,1,1} & + & (\mu_{1,2,1}+\mu_{1,2,2}) & + & (\mu_{1,3,1}+\mu_{1,3,2}+\mu_{1,3,3}) & + & \dots \\ 0 & & (\mu_{2,2,1}+\mu_{2,2,2}) & + & (\mu_{2,3,1}+\mu_{2,3,2}+\mu_{2,3,3}) & + & \dots \\ \vdots & & 0 & & (\mu_{3,3,1}+\mu_{3,3,2}+\mu_{3,3,3}) & + & \dots \\ & & \vdots & & 0 & & \\ & & & & \vdots & & \end{pmatrix} \\ &= \begin{pmatrix} \sum_{i=1}^n \mu_{1,i,1} & + & \sum_{i=2}^n \mu_{1,i,2} & + & \sum_{i=3}^n \mu_{1,i,3} & + & \dots \\ \sum_{i=2}^n \mu_{2,i,1} & + & \sum_{i=2}^n \mu_{2,i,2} & + & \sum_{i=3}^n \mu_{2,i,3} & + & \dots \\ \sum_{i=3}^n \mu_{3,i,1} & + & \sum_{i=3}^n \mu_{3,i,2} & + & \sum_{i=3}^n \mu_{3,i,3} & + & \dots \\ \vdots & & & & & & \end{pmatrix}\end{aligned}$$

to get to this final form

$$\boldsymbol{q}^M = \left(\sum_{j=1}^{n}\ddot{\theta}_j\cdot\sum_{k=max(j,i)}^{n}\operatorname{tr}\left(\frac{\partial W_k}{\partial\theta_i}J_k\frac{\partial W_k^T}{\partial\theta_j}\right)\right)_i. \tag{3.33}$$

For the $\mu_{i,j,k}$ holds

$$\begin{aligned}\mu_{i,j,k} &= \operatorname{tr}\left(\frac{\partial W_k}{\partial \theta_i} J_k \frac{\partial W_k^T}{\partial \theta_j}\right) \cdot \ddot{\boldsymbol{\theta}}_k = \operatorname{tr}\left(\frac{\partial W_k}{\partial \theta_i} J_k \frac{\partial W_k^T}{\partial \theta_j}\right)^T \cdot \ddot{\boldsymbol{\theta}}_k \\ &= \operatorname{tr}\left(\frac{\partial W_J}{\partial \theta_i} J_k^T \frac{\partial W_k^T}{\partial \theta_k}\right) \cdot \ddot{\boldsymbol{\theta}}_k \\ &= \mu_{k,j,i}\end{aligned}$$

which shows that $\mu_{i,j,k} = \mu_{k,j,i}$ as the inertial tensor J_k is a symmetric matrix. Because of this symmetry, the symmetric inertia matrix $M(\boldsymbol{\theta})$ can be obtained by:

$$M(\boldsymbol{\theta}) = \left(\sum_{k=max(j,i)}^{n} \operatorname{tr}\left(\frac{\partial W_k}{\partial \theta_i} J_k \frac{\partial W_k^T}{\partial \theta_j}\right)\right)_{i,j} \tag{3.34}$$

The partial derivatives of the partial forward kinematics W_j appear double in Eq. (3.34) and Eq. (3.31) in both $\boldsymbol{q}^M$ and $\boldsymbol{q}^C$ such that these terms cannot be natively modeled in the KBM form unlike the gravity component. As a consequence, the elements of the matrix M (and C which are very similar) are not combinations of trigonometric function like the kinematics but also contain their squared values. The square of a trigonometric functions basically means increasing its frequency to its double, for instance,

$$\cos^2 x = 1/2 \left(1 + \cos(2x)\right) \tag{3.35}$$

This can be observed in Figure 3.15 that displays the first component of the Coriolis matrix C for two degrees of freedom where the overlapping of the two frequencies becomes visible. Squared trigonometric functions cannot be expressed by means of a rational *quadratic* function. This problem can, however, be solved by raising the polynomial degree up to four. The main difficulty lies in obtaining the weights of the now five control vertices. Further, the index set has to be redefined for the increased number of $5^{d_{\text{in}}}$ control vertices.

Now assume that we have a rational Bézier curve $\boldsymbol{b}(\tau(\theta))$ with

$$b(\tau(\theta)) \equiv \sin(\theta)$$

with the associated weight vector $\boldsymbol{\gamma} = (1, \bar{\gamma} = \cos(\alpha), 1)$ for a chosen opening angle α. Then the squared values are obtained by

$$\begin{aligned}b^2(\underbrace{\tau(\theta)}_{=:t})^2 = sin^2(\theta) &= \left(t^2 + 2\bar{\gamma} \cdot t \cdot (1-t) + (1-t)^2\right)^2 \\ &= t^4 + \bar{\gamma} \cdot 4 \cdot t^3 (1-t) + \frac{1 + 2 \cdot \bar{\gamma}^2}{3} \cdot 6 \cdot t^2 (1-t)^2 \\ &=: \sum_{i=1}^{5} \gamma_i \cdot B_i^4(t), \quad \boldsymbol{\gamma} = (1, \bar{\gamma}, 1/3(1 + 2\bar{\gamma}^2), \bar{\gamma}, 1),\end{aligned}$$

The inertia term

$$\boldsymbol{q}^M = M(\boldsymbol{\theta}) \cdot \ddot{\boldsymbol{\theta}}$$

has a linear dependence on the joint accelerations in $\ddot{\boldsymbol{\theta}}$. The matrix cannot be separated from the the accelerations during learning such that they have to be combined with the rational basis function for learning

$$\ddot{\boldsymbol{\theta}}_i \cdot B^4_{\boldsymbol{i}}(\boldsymbol{\theta}_j), \quad \boldsymbol{i} \in \mathscr{I}_4^{d_{\text{in}}}.$$

This further increases the number of linear equations and related unknown parameters to

$$d_{\text{in}} \cdot 5^{d_{\text{in}}-1}.$$

With these modifications, it is possible to represent the elements of the inertia matrix M. This comes at high costs, however, as the minimal number of observations during training is increased drastically to $5^{d_{\text{in}}}$. The number can be slightly reduced, however, as it can be concluded that the generalized force in the first joint is not influenced by its joint position. The holds also for other joints and the total number of of training observations required can be reduced again to $5^{d_{\text{in}}-1}$.

Coriolis and centripetal coupling effects In analogy to the inertial terms, the Coriolis and centripetal coupling effects cannot be learned separated from the other components—the appropriate data can only be generated in simulation, or if the gravity component has already been identified and joints accelerations are set to zero. Their effects can be disassembled and prepared for the representation with KBM models in a way very similar to the inertia terms from Eq. (3.31)

$$q_i^C = \sum_{j=i}^{n} \sum_{k=1}^{j} \sum_{l=1}^{j} \underbrace{\left(tr\left(\frac{\partial \boxed{W_j}}{\partial \theta_i} J_j \frac{\partial^2 \boxed{W_j^T}}{\partial \theta_k \partial \theta_l} \right) \boxed{\dot{\theta}_k \dot{\theta}_l} \right)}_{\gamma_{i,j,k,l}}$$

and brought into this final form

$$\begin{aligned}
\boldsymbol{q}^C &= \begin{pmatrix} \gamma_{1,1,1,1} & + & \gamma_{1,2,1,1} + \gamma_{1,2,1,2} + \gamma_{1,2,2,1} + \gamma_{1,2,2,2} & \cdots \\ & & \gamma_{2,2,1,1} + \gamma_{2,2,1,2} + \gamma_{2,2,2,1} + \gamma_{2,2,2,2} & \cdots \\ & & \vdots & \end{pmatrix} \\
&= \left(\sum_{j=1}^{n} \sum_{k=1}^{n} \dot{\theta}_j \dot{\theta}_k \cdot \sum_{l=max(i,j,k)}^{n} tr\left(\frac{\partial W_l}{\partial \theta_i} J_l \frac{\partial^2 W_l^T}{\partial \theta_j \partial_k} \right) \right)_i \\
&=: C \cdot \left(\dot{\theta}_1 \dot{\theta}_1, \dot{\theta}_1 \dot{\theta}_2, \ldots, \dot{\theta}_1 \dot{\theta}_{d_{\text{in}}}, \dot{\theta}_2 \dot{\theta}_1, \dot{\theta}_2 \dot{\theta}_2, \ldots, \dot{\theta}_{d_{\text{in}}} \dot{\theta}_{d_{\text{in}}} \right)^T,
\end{aligned}$$

In this formula, the matrix C greatly resembles the matrix M of the inertia term $\boldsymbol{q}^M$ in the previous section. Again, the partial derivatives (but this time also second order derivatives) appear twice in the equation so that it can not be modeled with KBM but using the same rational form as the inertia matrix M. Another analogy is the additional linear dependence of the generalized coordinates. This time, however, the linear

	Degrees of freedom							
	1	2	3	4	5	6	7	8
Parameters	5	34	252	1,831	12,743	85,104	549,062	3,444,061

Table 3.1.: The number of model parameters of the KBM model when applied to the inverse dynamics in dependence of the number of degrees of freedom.

basis are the n^2 possible combinations of the joint velocities

$$\left(\dot{\theta}_1\dot{\theta}_1, \dot{\theta}_1\dot{\theta}_2, \ldots, \dot{\theta}_1\dot{\theta}_{d_{\text{in}}}, \dot{\theta}_2\dot{\theta}_1, \dot{\theta}_2\dot{\theta}_2, \ldots, \dot{\theta}_{d_{\text{in}}}\dot{\theta}_{d_{\text{in}}}\right)^T. \tag{3.36}$$

This vector contains redundancy as several combinations appear twice in its components (i.e., $\dot{\theta}_i\dot{\theta}_j$ equals $\dot{\theta}_j\dot{\theta}_i$ if $i \neq j$) and there remain only $\frac{d_{\text{in}}\cdot(d_{\text{in}}+1)}{2}$ unique combinations. After their elimination the rational form is completed with

$$\frac{d_{\text{in}}\cdot(d_{\text{in}}+1)}{2}\cdot 5^{d_{\text{in}}-1} \tag{3.37}$$

linear equations and unknown parameters.

Assembly of the components Every component of the equations of motion $\boldsymbol{q}^M$, $\boldsymbol{q}^C$ and $\boldsymbol{q}^g$ contributes a different number of linear equations and unknown parameters to the final assembled form. All equations are combined such that, if learned from examples generated by random movements, a total of

$$d_{\text{in}}\frac{d_{\text{in}}+3}{2}\cdot 5^{d_{\text{in}}-1} + 3^{d_{\text{in}}} \tag{3.38}$$

The number can be lowered by an offline exploration process where the training is separated into samples with zero acceleration and velocity for learning $\boldsymbol{q}^g$ and zero acceleration for $\boldsymbol{q}^C$. The inertia component $\boldsymbol{q}^M$ can be learned by subtracting the outputs of $\boldsymbol{q}^g$ and $\boldsymbol{q}^C$.

Altogether, this leads to a model of the inverse dynamics that is completely linear in its input variables. Analogously to the kinematics, such a model can be learned very efficiently and even exactly in simulation.

3.3. Summary

This chapter introduced the *Kinematic Bézier Maps (KBM)* a novel formalism to describe models of forward kinematics and inverse dynamics. Therefore, the reader first has been introduced to polynomial function approximation, Bézier techniques and projective geometry. The KBM have been developed on this foundation and provide a novel view on the representation of sensorimotor maps. In the following, the properties of the KBM were presented. The most intriguing of them is the linearization of the physical processes, that is, a formulation that is linear in model parameters. This opens the opportunity of the development of very efficient and exact learning algorithms which is presented in the following chapter.

4. Learning kinematic and dynamic models

In the chapter three, novel methods for the representation of kinematic and dynamic models have been presented. Their most prominent property is that they are linear in their model parameter because of their rational polynomial basis. This enables the design of very efficient learning algorithms. In this chapter, multiple algorithms related to the KBM representation will be presented that efficiently determine the parameters of the models.
The first section presents algorithms that find the solution that minimize the least mean square error over the training data. These methods perform batch learning that determines the model using a collection of training samples at a time. The next section introduces a learning rule for incremental learning with low computational complexity such that it can be used for online learning. Afterwards, a more sophisticated batch learning algorithm is presented that accounts for the (nonlinear) symmetry in the KBM representation in order to be less prone to overfitting to the training data. The last section demonstrates how the KBM models can be integrated into stochastic learning which allows modeling of additional nonlinear perturbations such as friction in the inverse dynamics.

4.1. Linear least mean squares

4.1.1. Normal equations

The search for the *linear least mean squares solution (LLMS)* is the most direct approach to learning forward models with the KBM. It minimizes the squared error over a set of training samples $\mathscr{T}$ with

$$\mathscr{T} = \left\{ (\boldsymbol{\theta}_i, \boldsymbol{x}_i) : i \in \{1, \ldots, m\},\ \boldsymbol{\theta}_i \in \Theta^{d_{\text{in}}},\ \boldsymbol{x} \in \mathbb{R}^{d_{\text{out}}} \right\}$$

for m observed training samples. The LLMS solution determines the model parameters of the KBM, that is, the set $\mathscr{C}$ that contains its $(3^{d_{\text{in}}})$ control vertices

$$\mathscr{C} = \{ \boldsymbol{b_i} : \boldsymbol{i} \in \mathscr{I}^{d_{\text{in}}} \},$$

such that the sum of the squared prediction errors over the training data, the so called *residues* r_i, becomes minimal. The standard linear least squares regression algorithm finds this solution with purely linear operations. Consequently, each output dimension d is treated separately as nonlinear operations such as the

inclusion of the Euclidean norm (which is quadratic) are not available. This leads to the following quadratic error function

$$e_d(\mathscr{C},\mathscr{T}) := \sum_{(\boldsymbol{\theta},\boldsymbol{x})\in\mathscr{T}} \Big(\sum_{\boldsymbol{i}\in\mathscr{I}^{d_{\text{in}}}} (b_{\boldsymbol{i},1}\cdot(B_{\boldsymbol{i}}\circ\tau_\alpha)(\boldsymbol{\theta})) - x_d \Big)^2 \tag{4.1}$$

$$= \boldsymbol{r}_d^T\cdot\boldsymbol{r}_d \tag{4.2}$$

which can be written as the product of the *residues vector* $\boldsymbol{r}$. The partial derivatives with respect to each coordinate of the control vertices are then

$$\frac{\partial e_d(\mathscr{C},\mathscr{T})}{\partial b_{\boldsymbol{i},d}} = 2\cdot((B_{\boldsymbol{i}}\circ\tau_\alpha)(\boldsymbol{\theta}_1),\ldots,(B_{\boldsymbol{i}}\circ\tau_\alpha)(\boldsymbol{\theta}_m))\cdot\boldsymbol{r}_d. \tag{4.3}$$

The global minimizer $\mathscr{C}^\star$ can then be determined by solving the error function in matrix notation

$$\mathscr{C}^\star = \arg\min_{\mathscr{C}} \sum_{d=1}^{d_{\text{out}}} e_d(\mathscr{C},\mathscr{T}) = \arg\min_{\mathscr{C}} \|C(\mathscr{C})\cdot B(\mathscr{T}) - X(\mathscr{T})\|_F$$

with

$$C(\mathscr{C}) = \Big(\boldsymbol{b}_{\iota^{-1}(i)}\Big)_{ij}$$
$$B(\mathscr{T}) = \Big((B_{\iota^{-1}(i)}\circ\tau_\alpha)(\boldsymbol{\theta}_j)\Big)_{i,j}$$
$$X(\mathscr{T}) = \Big(x_{j,i}\Big)_{i,j}, \quad (\boldsymbol{\theta}_i,\boldsymbol{x}_i)\in\mathscr{T}$$

where $C(\mathscr{C})$ is the matrix containing the control points in its columns as defined in Eq. (3.18), the matrix $B(\mathscr{T})$ contains the basis functions, and $X(\mathscr{T})$ has the cartesian coordinates of the training data in its columns. This minimization problem can be solved using the partial derivatives in Eq. (4.3) leading to a system of linear equations—the so called *normal equations*:

$$C(\mathscr{C})\cdot B(\mathscr{T})\cdot B(\mathscr{T})^T - X(\mathscr{T})\cdot B(\mathscr{T})^T = 0.$$

Its unique solution is a minimizer of the error function as $B(\mathscr{T})\cdot B(\mathscr{T})^T$ is positive definite. The LLMS solution is then given by

$$C(\mathscr{C}^\star) = X(\mathscr{T})\cdot B(\mathscr{T})^T\cdot\big(B(\mathscr{T})\cdot B(\mathscr{T})^T\big)^{-1} \tag{4.4}$$

where the minimizing control vertices $\boldsymbol{b}_{\boldsymbol{i}}^\star$ are contained in the matrix's columns. Here, the matrix

$$B(\mathscr{T})^+ = B(\mathscr{T})^T\cdot\big(B(\mathscr{T})\cdot B(\mathscr{T})^T\big)^{-1} \tag{4.5}$$

is called the *Moore-Penrose pseudo inverse* of $B(\mathscr{T})$. As the inverted matrix is always positive definite, the inversion can be facilitated by the Cholesky decomposition. If Eq. (4.1) is overdetermined, that is, there exist more training samples than model parameters for each dimension (i.e., $m \gg 3^{d_{\text{in}}}$), learning is capable of compensating a certain degree of sensor noise. If the noise is severe, however, the model still generalizes well in between data in the explored regions but differs greatly from the correct representation and produces very large prediction errors for completely unknown joint configurations. Furthermore, the

LLMS solution cannot natively perform incremental learning that is fast enough for online application as the complete training set has to be processed. While incremental learning can be emulated by increasing the sample pool $\mathscr{T}$ with new observations, the determination of the new model parameters is too expensive for higher degrees of freedom (usually for $d_{\text{in}} \geq 6$) and a 'forgetting' strategy has to be applied to control the number of elements in $\mathscr{T}$. Forgetting of older observations, for instance, is only desired if the kinematic structure changes but not for the refinement of the already learned model.

The following paragraphs will present a number of numeric solutions that overcome the limitation of the LLMS solution mainly with respect to incremental and online learning, and improved extrapolation. Yet, the LLMS solution is the first choice when learning from simulated data free of sensor noise as it has zero prediction errors over the whole configuration space of the robot. Such a model can be used as a starting point for the following numerical methods if an approximated forward model of the kinematics is available—which can be assumed to be the case in most applications.

4.2. Partial least squares

In the previous section, the standard linear least mean squares used for parameter detection has been presented. For a high number of degrees of freedom, this approach becomes intractable. This is because of inversion of the matrix containing the rational polynomial basis whose number of columns increases exponentially with the number of degrees of freedom—and with it the required memory for the inversion. There exist many numerical methods that address this problem such as factorizations or *recursive least squares*. In this thesis, the one-dimensional *partial least squares (PLS-1)* method has been studied. It is also applied in the *locally weighted projection regression (LWPR)* (see Section 2.3.4). This is, on the one hand, for the avoidance of matrix inversions and, on the other hand, for the ability to detect irrelevant and correlated inputs.

4.2.1. The algorithm

This paragraph describes the partial least squares algorithm for a single output dimension (PLS-1). Note that this algorithm learns each output dimension separately similar to the naive LLMS. For this reason, an univariate output is considered in this section (for instance, only the x-coordinate of the end-effector). The PLS-1 algorithm transforms the linear regression problem into a sequence of parallel projections that transform the input space into one-dimensional subspaces where the regression can be solved easily. The maximal number of projections equals the number of input variables in the system of linear equations. In case of the KBM, this means $3^{d_{\text{in}}}$—the number of rational basis polynomials. In each step $k \leq 3^{d_{\text{in}}}$, the input data is projected along a vector $\boldsymbol{u}_k$ that reflects the linear correlation between the input and output data. It is proportional to the covariance:

$$\boldsymbol{u}_k \propto \Big(\mathrm{COV}\big((B_{\iota^{-1}(i)} \circ \tau_\alpha)(\boldsymbol{\theta}_j), x_j\big)\Big)_{i,j} \in \mathbb{R}^{3^{d_{\text{in}}} \times n},$$

Algorithm 6: Pseudo code for learning the KBM with partial least squares (PLS-1)

Input: n samples in $\mathscr{T} = \{(\boldsymbol{\theta}_i, x_i) : i \in \{1,\ldots,n\}\}$, number of projections m
Output: Parameters of the partial least squares $\boldsymbol{p}_k, q_k, u_k : \forall k \in \{1,\ldots,m\}$

$W_0 := \left(B_{\tau(i)} \circ \tau)(\boldsymbol{\theta}_j)\right)_{i,j} \in \mathbb{R}^{3^{d_{\text{in}}} \times n}, \quad \boldsymbol{x} \in \mathbb{R}^{n \times 1}$
for $k = 1 : m \leq 3^{d_{in}}$ **do**
  $\boldsymbol{u}_k \in \mathbb{R}^{3^{d_{\text{in}}}} := W_{k-1} \cdot \boldsymbol{x}_{k-1}$
  $\boldsymbol{b}_k := W_{k-1}^T \cdot \boldsymbol{u}_k \in \mathbb{R}^{n \times 1}$
  $q_k := \frac{\boldsymbol{x}^T \cdot \boldsymbol{b}_k}{\|\boldsymbol{b}_k\|^2}$
  $\boldsymbol{x}_k := \boldsymbol{x}_{k-1} - q_k \cdot \boldsymbol{b}_k$
  $\boldsymbol{p}_k := \frac{B_{k-1} \cdot \boldsymbol{b}_k}{\|\boldsymbol{b}_k\|^2} \in \mathbb{R}^{3^{d_{\text{in}}}}$
  $W_k := W_{k-1} - \boldsymbol{p}_k \cdot \boldsymbol{b}_k^T$
  print $e_k := \Sigma_i^n \|x_{k,i}\| / n$

for a given training set

$$\mathscr{T} = \{(\boldsymbol{\theta}_i, x_i) : i \in \{1,\ldots,n\}\}.$$

This distinguishes the algorithm from the *principal component regression (PCR)* which projects the input on the one-dimensional subspaces defined by the eigenvectors of the covariance matrix of the inputs. While this procedure finds redundancies in the input, it completely neglects the influence on the output values. Consequently, the importance of the input variables—or their irrelevance—cannot be identified.

In the first step of the PLS-1, the input and output data W_0 and $\boldsymbol{x}_0$ are defined as

$$W_0 := \left(B_{\iota^{-1}(i)} \circ \tau_\alpha)(\boldsymbol{\theta}_j)\right)_{i,j} \in \mathbb{R}^{3^{d_{\text{in}}} \times n}, \quad \boldsymbol{x}_0 = \left(x_1,\ldots,x_n\right)^T \in \mathbb{R}^n. \tag{4.6}$$

The first projection vector is then defined as

$$\boldsymbol{u}_1 := W_0 \cdot \boldsymbol{x}_0, \tag{4.7}$$

and reflects the correlation between the input data W_0 and output data $\boldsymbol{x}_0$. Note that it is not necessary to subtract the mean values from the data as this is performed implicitly by the algorithm,.

Then the input data is projected along $\boldsymbol{u}_1$

$$\boldsymbol{b}_1 := W_0^T \cdot \boldsymbol{u}_1,$$

where the *vector* $\boldsymbol{b}_1$ is the one-dimensional image of W_0. Now a linear regression can be performed over the one-dimensional $\boldsymbol{b}_1$ resulting in the first regression factor q_1:

$$q_1 = \arg\min_q (\boldsymbol{x}_0 - q \cdot \boldsymbol{b}_1)^2 = \frac{\boldsymbol{x}_0^T \cdot \boldsymbol{b}_1}{\|\boldsymbol{b}_1\|^2}. \tag{4.8}$$

Then, the residue in the output value after the regression is given by

$$\boldsymbol{x}_1 := \boldsymbol{x}_0 - q_1 \cdot \boldsymbol{b}_1,$$

resulting in the error

$$e_1 = \frac{\sum_i^n |x_{i,1}|}{n}.$$

The residues of the input values are calculated by another univariate linear regression to determine the projection vector $\boldsymbol{p}_1$:

$$\boldsymbol{p}_1 = \arg\min_{\boldsymbol{p}} \left(W_0 - \boldsymbol{p} \cdot \boldsymbol{b}_1^T\right)^2 = \frac{W_0 \cdot \boldsymbol{b}_1}{\|\boldsymbol{b}_1\|^2} \in \mathbb{R}^{3^{d_{\text{in}}}}$$

leading to

$$W_1 := W_0 - \boldsymbol{p}_1 \cdot \boldsymbol{b}_1^T$$

Now, the algorithms repeats these steps starting with Eq. (4.7) until either the error e_k falls below a threshold or k surpasses a maximal number of $k_{\max}$ steps or the number of input dimensions (that is, $k_{\max} = 3^{d_{\text{in}}}$). The complete algorithm is shown in Algorithm. 6.

4.2.2. Learning KBM models with partial least squares

In this paragraph, it is shown how the partial least squares algorithm can be applied together with the KBM models.

Reconstruction of the control net

Once the parameters of the partial least squares regression $\boldsymbol{p}_k, q_k, \boldsymbol{u}_k$ and $k_{\max}$ have been determined where $1 \leq k \leq k_{\max} \leq 3^{d_{\text{in}}}$, the control net $\boldsymbol{c}$ of the KBM can be reconstructed. Therefore, the weight matrix W_0 is initialized as the identity matrix and the starting value $\boldsymbol{c}_0$ for the control net as the zero vector

$$W_0 := \begin{pmatrix} 1 & & 0 \\ & \ddots & \\ 0 & & 1 \end{pmatrix} \in \mathbb{R}^{3^{d_{\text{in}}} \times 3^{d_{\text{in}}}}, \qquad \boldsymbol{c}_0 := (0, \ldots, 0)^T \in \mathbb{R}^{n \times 1}.$$

The reconstruction is similar to the original algorithm. In the first step, the projection $\boldsymbol{b}_1$ of W_0 is computed

$$\boldsymbol{b}_1 = W_0^T \cdot \boldsymbol{u}_1,$$

and the regressions in Eq. (4.8) and Eq. (4.8) are reverted

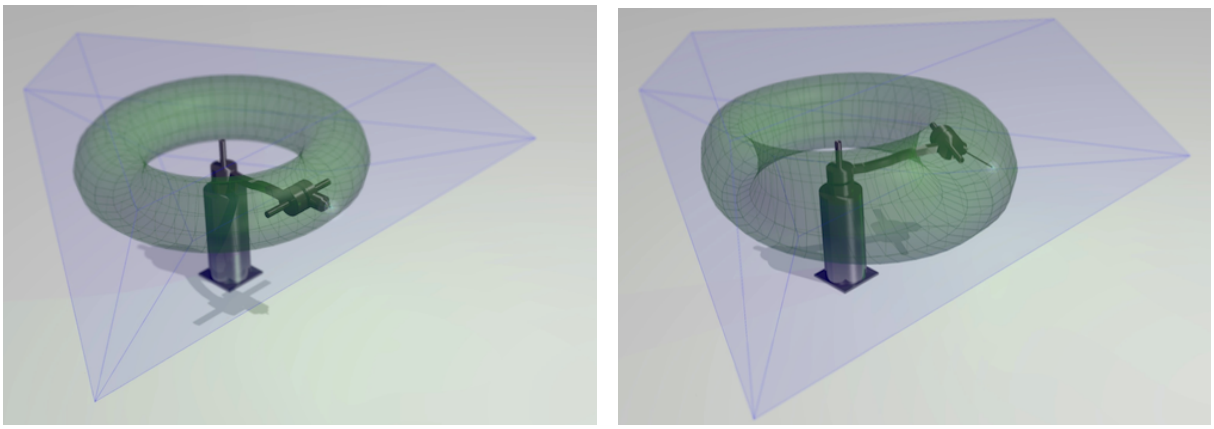
$$\boldsymbol{c}_1 = \boldsymbol{c}_0 + q_1 \cdot \boldsymbol{b}_1 \text{ and } W_1 = W_0 - \boldsymbol{p}_1 \cdot \boldsymbol{b}_1^T.$$

These calculations are repeated $k_{\max} \leq 3^{d_{\text{in}}}$ times resulting in the final net stored in $\boldsymbol{c}_{k_{\max}}$. The complete algorithm is depicted in Algorithm. 7. Once all dimensions of the control net are reconstructed, the net can be used for prediction and incremental learning (see Section 4.3).

Algorithm 7: Pseudo code for the reconstruction of the control net.

Input: Parameters of the partial least squares: $\boldsymbol{p}_k, q_k, u_k : k \in \{1, \ldots, m\} n$, and the number of steps $k_m ax$
Output: Control net in $\boldsymbol{c}_m \in \mathbb{R}^{3^{d_{\text{in}}}}$

$W_0 \leftarrow$ identity matrix $\in \mathbb{R}^{3^{d_{\text{in}}} \times 3^{d_{\text{in}}}}$
$\boldsymbol{c}_0 \leftarrow (0, \ldots, 0)^T \in \mathbb{R}^{n \times 1}$
for $k = 1 : k_{\max} \leq 3^{d_{in}}$ **do**
    $\boldsymbol{b}_k = W^T \cdot \boldsymbol{u}_k$
    $\boldsymbol{c}_k = \boldsymbol{c}_{k-1} + q_k \cdot \boldsymbol{b}_k$
    $W_k = W_{k-1} - \boldsymbol{p}_k \cdot \boldsymbol{b}_k^T$



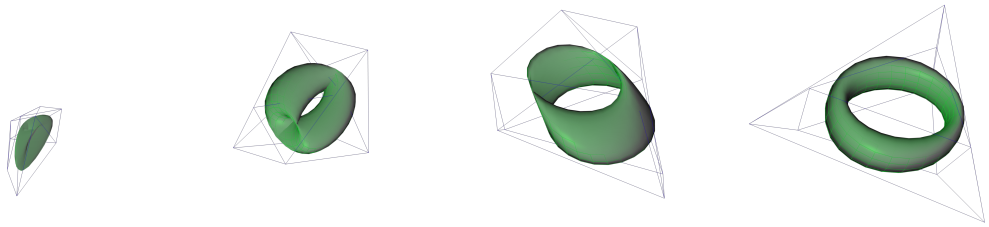
(a) Exact model.

(b) Adaptation to wielding a tool.

Figure 4.1.: Kinematic Bézier map model for a robot with two orthogonal axes.

Properties of the learned model

Learning with the partial least squares algorithm is an enormous improvement compared to learning with the naive LLMS (see Section 4.1). First, this is because the algorithm does not invert the weight matrix W_0 in Eq. (4.6) opposed to the normal equations in Eq. (4.5). For many input dimensions (which grow exponentially with the number of controllable degrees of freedom), this matrix inversion requires a large amount of working memory ($\mathcal{O}(3^{d_{\text{in}} \cdot 3})$) rendering the method intractable for higher numbers of degrees of freedom. While other numerical approaches such as matrix decompositions and recursive least squares are faster, the PLS-1 regression offers another interesting property: When applied to LWPR for instance, it is able to detect redundant and less significant input dimensions. While this cannot be exploited with the KBM model (as each rational basis polynomial depends on all joint angles), the algorithm can terminate after a mean prediction error reaches a threshold. If this threshold equals the (a priori known) sensor noise on a real system or in simulation, the algorithm effectively prevents overfitting to the noisy data. That way, the model can be learned from a significantly lower amount of training data while maintaining a higher prediction accuracy.



(a) 5 samples (b) 10 samples (c) 20 samples (d) 40 samples

Figure 4.2.: Sequence showing the online learning process of a KBM at different numbers of learned samples. Unlike in the previous experiments, these samples are uniformly distributed over the whole parameter space. *Source: (Ulbrich et al., 2012c)*

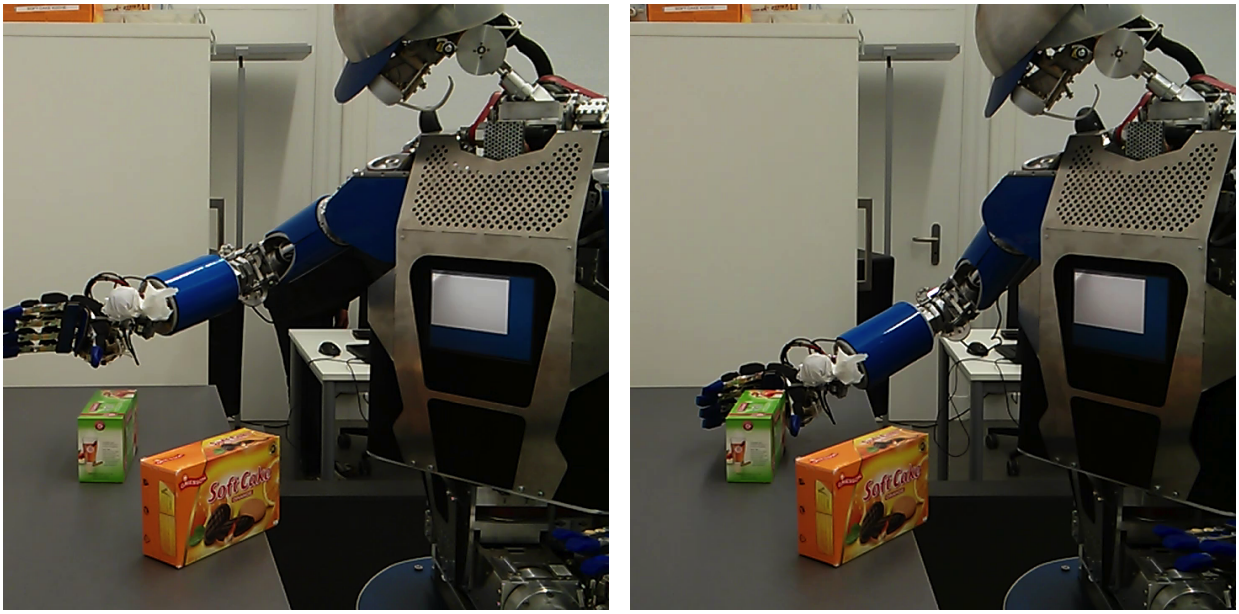


Figure 4.3.: The humanoid robot ARMAR-III grasping an object. During training with incremental learning, the robot grasped the object using visual servoing. The KBM has been initialized to the inbuilt forward kinematics. Afterwards, the marker for the detection of the hand had been covered. The robot then had to predict the correct position of the hand to repeat the grasp without being able to recognize its hand in the image. In the picture on the left, the robot relied only on the inbuilt, inaccurate forward kinematics. The right picture shows the reproduction of the grasp using the incrementally learned model.

4.3. The δ-Rule

The δ-rule is probably the most interesting and useful algorithm for learning a KBM model. This is, on the one hand, because of the generation of models with good extrapolation and, on the other hand, because of the simplicity of the algorithm and its very low computational costs that make online application possible even for long kinematic chains. The δ-rule is also known as *Hoff-Widrow-rule* named after its inventors (Widrow and Hoff, 1960) or *(normalized) least mean squares (NLMS)*. Originally, it is a steepest descent-based training methods that allows incremental adaptation of the model parameters and is usually applied to learning single layer preceptrons (Widrow and Hoff, 1960). From its incremental nature, results the main advantage of this rule: A robot using this learning rule can perform online and life-long adaptation to its kinematic structure and dynamics. Last but not least, it is neurologically inspired and effective despite keeping its

simplicity which allows it to be easily integrated and applied on humanoid robots. Figure 4.2 show the learning the surface of a torus with the δ-rule and Figure 4.3 shows the application on a the humanoid robot ARMAR-III.

4.3.1. The algorithm

The algorithm adapts the model parameters (i.e., control vertices) incrementally for each observed training sample $(\boldsymbol{\theta}_i, \boldsymbol{x}_i)$ from a data set

$$\mathscr{T} = \{(\boldsymbol{\theta}_i, \boldsymbol{x}_i) : i \in \mathbb{N}\}.$$

As it processes only one single sample during an increment, the data set can be of infinite size, that is, the robot can perform life-long learning. The method can coexist with visual servoing, continuously learning when the end-effector is visible and predicting configurations similar to those experienced before (see Figure 4.3). Similar to the analytic linear least mean squares solution, the control net is trained for each output dimension separately. Therefore, an error function $e_j(\cdot)$ with respect to each component of each output dimension j is created in each step in each step i:

$$e_j(\mathscr{C}, (\boldsymbol{\theta}_i, \boldsymbol{x}_i)) = \left(\left(\sum_{\boldsymbol{i} \in \mathscr{I}^{d_{\text{in}}}} b_{\boldsymbol{i},j} \cdot (B_{\boldsymbol{i}} \circ \tau_\alpha)(\boldsymbol{\theta}_i) \right) - \boldsymbol{x}_i \right)^2 =: \boldsymbol{r}_i^T \cdot \boldsymbol{r}_i, \quad j = 1, \ldots, d_{\text{out}}$$

Its partial derivatives are given by

$$\frac{\partial e_j(\mathscr{C}, (\boldsymbol{\theta}_i, \boldsymbol{x}_i))}{\partial b_{\boldsymbol{i},j}} = (B_{\boldsymbol{i}} \circ \tau_\alpha)(\boldsymbol{\theta}_i) \cdot r_{i,j}.$$

The direction of the steepest descent is then

$$\nabla e_j(\mathscr{C}, (\boldsymbol{\theta}_i, \boldsymbol{x}_i)) = \left(\frac{\partial e_j(\mathscr{C}, (\boldsymbol{\theta}_i, \boldsymbol{x}_i))}{\partial b_{\iota(i),j}}, \ldots, \frac{\partial e_j(\mathscr{C}, (\boldsymbol{\theta}_i, \boldsymbol{x}_i))}{\partial b_{\iota(i),j}} \right)_i$$

and the control net is updated by the rule

$$\left(b_{\iota(i),j}^{(i+1)} \right)_{i \in \{1, \ldots, 3^{d_{\text{in}}}\}} = \left(b_{\iota(i),j}^{(i)} \right)_{i \in \{1, \ldots, 3^{d_{\text{in}}}\}} - \nu \cdot \nabla e_j(\mathscr{C}, (\boldsymbol{\theta}_i, \boldsymbol{x}_i)), \nu \in \mathbb{R}$$

such that each component of the control vertices is decremented by its partial derivative of the error function multiplied by the learning rate ν. Instead of determining ν manually (e.g., by line search), the following form, which is known as *normalized* least mean squares (NLMS), can be used

$$\left(b_{\iota(i),j}^{(i+1)} \right)_{i \in \{1, \ldots, 3^{d_{\text{in}}}\}} := \left(b_{\iota(i),j}^{(i)} \right)_{i \in \{1, \ldots, 3^{d_{\text{in}}}\}} - \tilde{\nu} \cdot \frac{\nabla e_j(\mathscr{C}, (\boldsymbol{\theta}_i, \boldsymbol{x}_i)) \cdot e_j(\mathscr{C}, (\boldsymbol{\theta}_i, \boldsymbol{x}_i))}{\nabla e_j(\mathscr{C}, (\boldsymbol{\theta}_i, \boldsymbol{x}_i))^T \cdot \nabla e_j(\mathscr{C}, (\boldsymbol{\theta}_i, \boldsymbol{x}_i))}, \quad 0 < \tilde{\nu} \leq 1 \quad (4.9)$$

where the learning rate can be chosen in a simpler way. A learning rate with $\nu = 1$ means that the model predicts exactly the observed cartesian coordinate $\boldsymbol{x}_i$ after the increment. The computational costs for the

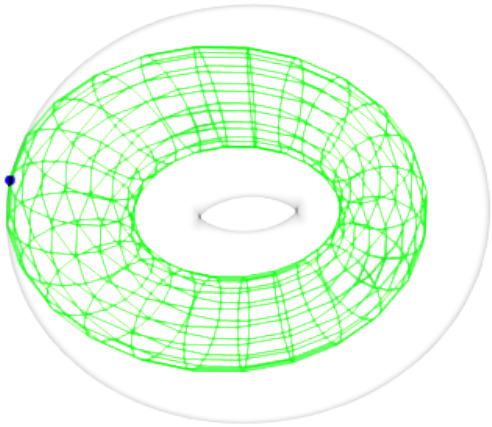


Figure 4.4.: The impact of learning a single sample with the incremental learning after a kinematic change. A perfect approximation of an exemplary workspace of a robot with two degrees of freedom (learned with batch learning in simulation) adapts to a change where the size of the last link has been doubled. The new workspace is rendered transparently. When learning from a single example with the incremental learning algorithm, the manifolds directly interpolates the new data point but does not influence the rest of the representation significantly, especially at the far end of the configuration space.

adaptation to a single observation are consequently only marginally higher than the costs for the prediction of the output value. This is why the algorithm can be applied online on robots with many independent degrees of freedom and enable continuous, that is, life-long, learning. The same training samples can be processed multiple times. Later in the evaluation (Section 6.3), this is referred to as *micro batch* learning, a fixed number of samples is then repeatedly processed.

Compared to the LLMS solution presented in the previous section, the δ-rule requires significantly more training data and converges slower to the desired solution when learning from scratch. With sensor noise, this is, however, an advantage and learning can be stopped when the predictions over a representative data set reach the expected accuracy.

4.3.2. Initialization

Incremental methods such as the δ-rule and the nonlinear optimization presented afterwards benefit from a good start value which drastically speeds up the learning procedure (i.e., reduces the number of training samples). Ideally, an approximative forward model of the dynamics or kinematics of a robot is available or can be generated easily (see Figure 4.3).

From Eq. (4.9) it can be seen that the control vertices are updated into direction of the prediction error $b(\boldsymbol{\theta}_i) - \boldsymbol{x}_i$ weighted by their polynomial basis. As a consequence, adaptation to a single sample has mainly a local impact on the model (see Figure 4.4). The prediction for joint configurations that lie far apart—including exteroception—are not significantly influenced. The model hence does not "forget" the approximate model it has been initialized with such that the prediction errors for the extrapolation remains close to

those of the approximate model. A model generated with the LLMS algorithm suffers from a significantly higher extrapolation errors due to overfitting when the data is noisy and not enough redundant data can be learned from.

However, if no a priori model is available at all, the KBM model can be initialized by a linearized model obtained by batch learning. Therefore, the kinematics is approximated by a linear model. This requires the training samples to be generated with small joint angles around zero (for instance, $|\theta_{j,i}| < 15°$). Then, a linear approximation

$$l(\cdot): \qquad \mathbb{R}^{d_{\text{in}}} \to \mathbb{R}^{d_{\text{out}}}, \qquad \boldsymbol{\theta} \mapsto L \cdot \begin{pmatrix} \theta_1, \ldots, \theta_{d_{\text{in}}}, 1 \end{pmatrix}^T, L \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$$

can be obtained by solving

$$L \cdot \begin{pmatrix} & \theta_{j,i} & \\ 1 & \ldots & 1 \end{pmatrix}_{j,i} = \begin{pmatrix} & x_{j,i} & \\ 1 & \ldots & 1 \end{pmatrix}_{i,j} \tag{4.10}$$

for the matrix L. The control vertices of the initial net for a given spread angle α are set according to their index vector $\boldsymbol{i}$ as

$$\boldsymbol{b_i} = L \cdot \boldsymbol{\rho_i}$$

$$\boldsymbol{\rho_i} = \begin{pmatrix} \rho_1, \ldots, \rho_{d_{\text{in}}}, 1 \end{pmatrix}^T, \text{ where}$$

$$\rho_j = \begin{cases} -\alpha, & i_j = 0 \\ 0, & i_j = 1 \\ \alpha, & i_j = 2 \end{cases}, \quad \boldsymbol{i} = \begin{pmatrix} i_1, \ldots, i_{d_{\text{in}}} \end{pmatrix} \in I^{d_{\text{in}}}.$$

The pseudocode for this linear initialization is given in Algorithm. 8 and a picture of the initialization for three degrees of freedom is shown in Figure 4.5. Although the model is obtained by batch-learning, only $d_{\text{in}} + 1$ angles are at least required due to the linearity of the model. This initialization can greatly speed up learning albeit not being by far as effective as starting with an approximate model as described above.

Algorithm 8: Optimal starting value for numeric approaches.

Input: Training data set $\mathscr{D} = \{(\boldsymbol{\theta}_i, \boldsymbol{x}_i) : i = 1, \ldots, m\}$, Opening angle δ
Requirements : Angles $|\boldsymbol{\theta}_i| \lesssim \delta$
Output: Initial control net C^0

`// Convert training data to matrices. Data is stored in the matrices' columns.`
$P \in \mathbb{R}^{d_{\text{in}} \times m}$ `// Proprioception`
$S \in \mathbb{R}^{d_{\text{out}} \times m}$ `// Exteroception`
$P := ((\boldsymbol{\theta}_i)_j)_{ji}$ and $S := ((\boldsymbol{x}_i)_j)_{ji}$
Solve $L \cdot \begin{pmatrix} P \\ 1 \end{pmatrix} = \begin{pmatrix} S \\ 1 \end{pmatrix}$ for linear model $L \in \mathbb{R}^{m \times d_{\text{in}}}$
$\mathscr{C}_0 = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_{3^{d_{\text{in}}}}\}$ `// Define initial control net`
for $i = 1 : 3^{d_{\text{in}}}$ **do** `// initialize control vertices`
    $\boldsymbol{c}_i = L \cdot (\iota^{d_{\text{in}}}(i) - (1, \overset{d_{\text{in}}}{\ldots}, 1)) \cdot \delta$

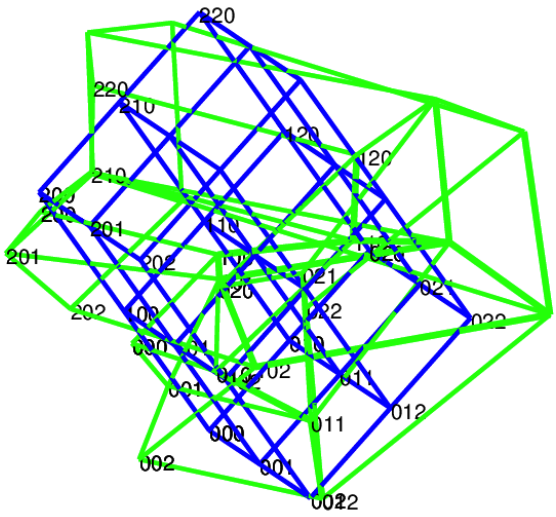


Figure 4.5.: Initialization of a KBM's control net by linearization of the training data.

4.4. Approximation with nonlinear symmetry constraints

The linear optimization techniques presented in the previous two sections are not capable of taking the symmetric and regular structure of a well-formed control net of the KBM representation into account (see Figure 3.9). While such a structure emerges spontaneously in most cases this does not occur under aggravated conditions. Especially if the joint angles generating the training data lie close together and are subject to sensor noise, not enough information can be generalized from the data and the control net degenerates *(overfitting)*. As a consequence, the KBM model loses its ability of extrapolation, that is, to reliably predict the location of its limbs for joint configurations not observed during training. This can be explained by the fact that the extrapolation consists of non-convex operations which means that smaller errors in interpolation lead to large errors in extrapolation.

One way to address this problem is the consideration of the desired symmetric structure to constrain the solution found by the parameter determination. Although this leads to a more reasonable representation, these constraints include nonlinear error terms which greatly complicate learning and increase computation time. On the other hand, it offers very interesting insights on the regular and symmetric structures in this sensorimotor representation. The early ideas underlying this research were presented in (Ulbrich and Asfour, 2013).

4.4.1. Symmetry and regularity in the control net

Before regularity and symmetry constraints can be formulated, a closer look at the geometric nature of a well-formed control net has to be taken. It comprises a very regular structure where many connected vertices form isosceles triangles with a predefined common angle (see Eq. (3.19)). This does not hold for every triangle though, so that they have to be identified and counted by induction.

Given a kinematics $k(\boldsymbol{\theta})$ with d_{in} identified rotation axes and given that $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_{d_{\text{in}}})$ is a joint con-

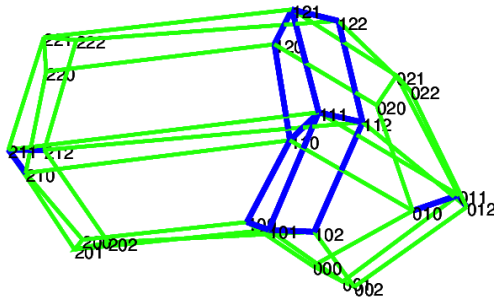


Figure 4.6.: Symmetry in an arbitrary kinematics with three degrees of freedom. Green lines belong to isosceles triangles with the common angle α. Blue lines indicate asymmetric triangles.

figuration where the angles are sorted by their lateral distance to the kinematics origin, the position of the end-effector is defined by a function

$$k(\cdot): \qquad \Theta^{d_{\text{in}}} \to \mathbb{R}^3, \qquad \boldsymbol{\theta} \mapsto M_1(\theta_1) \cdot M_2(\theta_2) \cdot \ldots \cdot M_{d_{\text{in}}}(\theta_{d_{\text{in}}}) \cdot \boldsymbol{m},$$

where the $M_i \in SE_3$ are rigid body transformations depending on a single joint angle θ_i each and $\boldsymbol{m}$ is the final transformation from the last axis to the end-effector. The matrix M_1 represents the most *proximal* and $M_{d_{\text{in}}}$ the most *lateral* degree of freedom.

The control net can then be constructed manually in the following way: An initial control net $\mathscr{C}_0$ is defined as the set containing a single point

$$\mathscr{C}_0 = \{k(\boldsymbol{\theta})\}$$

Let $\boldsymbol{r}_1$ be the last (i.e., most lateral) rotation axis for a configuration $\boldsymbol{\theta}$. For this degree of freedom, a control net $\mathscr{C}_i$ can be constructed by a scaling $S_{\boldsymbol{r}_1}$ orthogonal to $\boldsymbol{r}_1$ and two rotations $R_{\boldsymbol{r}_1}(\alpha)$ and $R_{\boldsymbol{r}_1}(-\alpha)$ around $\boldsymbol{r}_1$ where α is the chosen opening angle of the net. This procedure is repeated for all the other axes approaching the kinematics' origin resulting in the inductive rule

$$\mathscr{C}_{i+1} = R_{\boldsymbol{r}_i}(\alpha) \cdot \mathscr{C}_i \cap S_{\boldsymbol{r}_i} \cdot \mathscr{C}_i \cap R_{\boldsymbol{r}_i}(-\alpha) \cdot \mathscr{C}_i, \tag{4.11}$$

where the product of a transformation and a set here means that every single element is transformed. In each step, every control vertex becomes connected to a new neighbor which is reflected in the indices in Eq. (3.19). These neighbors form symmetric triangles. The scalings $S_{\boldsymbol{r}_i}$, however, do not preserve this symmetry in general such that the number of isosceles triangles Eq. (3.19) can be derived inductively by

$$m_{i+1} = 2 \cdot m_i + |\mathscr{C}_i|, \text{ where } |\mathscr{C}_i| = 3^i. \tag{4.12}$$

This results in a total of $m_{d_{\text{in}}} = 3^{d_{\text{in}}} - 2^{d_{\text{in}}}$ which can be proven by induction:

$$\begin{aligned} m_1 &= 3^1 - 2^1 = 1, \\ m_{i+1} &= 2 \cdot (3^i - 2^i) + 3^i = 3^{i+1} - 2^{i+1} \quad \square \end{aligned}$$

A triangle related to the d-th degree of freedom is consequently isosceles if and only if for the indices of all

its control vertices $\boldsymbol{b_i}$ holds

$$\boldsymbol{i} = (i_1, \ldots, i_d, \ldots i_{3^{d_{\text{in}}}}) \Rightarrow i_j \neq 1, \forall j < d.$$

In the two-dimensional case, for instance, the only non-isosceles triangle is

$$\boldsymbol{b}_{(1,0)}\boldsymbol{b}_{(1,1)}\boldsymbol{b}_{(1,2)}.$$

Figure 4.6 shows this regular structure of a three-dimensional KBM model and highlights the non-symmetry (eight triangles) in blue color. In order to access the (not necessarily isosceles) triangles that belong to a control point, the index function $\bar{\iota}$ defined in Section 3.2.1 is required.

Algorithm 9: Pseudo code for the creation of a lookup table reverse index function $\tilde{\iota}^{-1}$ that is used for the creation of the Jacobian matrix and the residues vector.

Input: Index function $\iota(\cdot) : \mathbb{N}^+ \rightarrow \mathscr{I}^{d_{\text{in}}}$
Output: Lookup table I for $\tilde{\iota}^{-1} \in \mathbb{N}^{3^{n_{DoF}} \times n_{DoF}}$

for $i = 1 : 3^{d_{in}}$ **do**
  for $j = 1 : d_{in}$ **do**
    $\boldsymbol{i} = (i_1, \ldots, i_{d_{\text{in}}}) := \iota(j)$
    $\tilde{\boldsymbol{i}} := (i_1, \ldots, \not{i_j}, \ldots, i_{d_{\text{in}}})$
    $I_{i,k} = (j-1) \cdot 3^{n_{DoF}-1} + \sum_k \left(3^k \cdot \tilde{i}_k\right) + 1$

4.4.2. Formulation of the error terms

The nonlinear optimization requires an error function to be minimized. The real-valued error function that minimizes the prediction error over the training data and constrains the symmetry is a combination of three error functions—one for the isosceles property e_Δ, one for the opening angle e_α and one for the prediction

Algorithm 10: Optimization of the leg length in the isosceles triangles.

Input: index $\boldsymbol{i} \in \mathbb{R}^{3^{n_{DoF}} \times n_{DoF}}$
Output: J_Δ,r_Δ

for $i = 1 : 3^{n_{DoF}}$ **do** `/* no comment */`
  for $j = 1 : n_{DoF}$ **do**
    $\boldsymbol{a} = CP_{rIndex_{i,j,1}}$
    $\boldsymbol{b} = CP_{rIndex_{i,j,2}}$
    $\boldsymbol{c} = CP_{rIndex_{i,j,3}}$
    $v = index_{i,\star}$
    $(J_\Delta)_{\bar{\iota}^{-1}(i,k),i} = \begin{cases} 2 \cdot (\boldsymbol{a} - \boldsymbol{b}), & v_j = 1 \\ 2 \cdot (\boldsymbol{c} - \boldsymbol{a}), & v_j = 2 \\ 2 \cdot (\boldsymbol{b} - \boldsymbol{c}), & v_j = 3 \end{cases}$
    $(r_\Delta)_{\bar{\iota}^{-1}(i,k)} = \|a - b\|^2 - \|b - c\|^2$

Algorithm 11: Optimization of the common angle in the isosceles triangles.

Input: index $\in \mathbb{R}^{3^{n_{DoF}} \times n_{DoF}}$
Output: J_α,r_α

for $i = 1 : 3^{n_{DoF}}$ **do**
  for $j = 1 : n_{DoF}$ **do**
    $\boldsymbol{a} = CP_{rIndex_{i,j,1}}$
    $\boldsymbol{b} = CP_{rIndex_{i,j,2}}$
    $\boldsymbol{c} = CP_{rIndex_{i,j,3}}$
    $v = index_{i,\star}$
    $(J_\alpha)_{\rho(i,k),i} = \begin{cases} \boldsymbol{a}+\boldsymbol{c}-2\cdot\boldsymbol{b}-\tan(\delta)^2/2\cdot(\boldsymbol{a}-\boldsymbol{c}), & v_j = 1 \\ 2\cdot\boldsymbol{b}-(\boldsymbol{a}+\boldsymbol{c}), & v_j = 2 \\ \boldsymbol{a}+\boldsymbol{c}-2\cdot\boldsymbol{b}-\tan(\delta)^2/2\cdot(\boldsymbol{a}-\boldsymbol{c}), & v_j = 3 \end{cases}$
    $(r_\alpha)_{\rho(i,k)} = \|1/2(\boldsymbol{a}+\boldsymbol{c})-\boldsymbol{b}\|^2 - 1/4\tan^2(\alpha)\cdot\|1/2(\boldsymbol{a}-\boldsymbol{c})\|^2$

error $e_{\mathscr{T}}$ over the training data $\mathscr{T} = \{(\boldsymbol{\theta}_i, \boldsymbol{x}_i)\}$ respectively which all depend on the model parameters, that is, the set of control vertices $\mathscr{C}$

$$e(\mathscr{C},\mathscr{T}) = \omega_1\cdot e_\Delta(\mathscr{C}) + \omega_2\cdot e_\alpha(\mathscr{C}) + \omega_3\cdot e_{\mathscr{T}}(\mathscr{C},\mathscr{T}) \tag{4.13}$$

$$= \omega_1\cdot \boldsymbol{r}_\Delta(\mathscr{C})^T\cdot\boldsymbol{r}_\Delta(\mathscr{C}) + \omega_2\cdot\boldsymbol{r}_\alpha(\mathscr{C})^T\cdot\boldsymbol{r}_\alpha(\mathscr{C}) \tag{4.14}$$

$$+ \omega_3\cdot\boldsymbol{r}_{\mathscr{T}}(\mathscr{C},\mathscr{T})^T\cdot\boldsymbol{r}_{\mathscr{T}}(\mathscr{C},\mathscr{T}).$$

Each error function itself is a squared sum of residues pooled in the *residue vectors* $\boldsymbol{r}_\square(\mathscr{C})$. The optimization process finds an optimal control net $\mathscr{C}^\star$ for which holds

$$\mathscr{C}^\star = \arg\min_{\mathscr{C}} e(\mathscr{C},\mathscr{T}) \tag{4.15}$$

Prediction error over the training data

There are two ways of defining the error function $e_{\mathscr{T}}(\mathscr{C},\mathscr{T})$. The first is very similar to the nonlinear optimization with the only difference that it includes the errors in every output dimension d_{out} whereas those were previously treated separately. This results in a $(d_{\text{out}}\cdot n)$-sized vector for n training samples. Let $\boldsymbol{e}_i$ be the prediction error for the training sample $(\boldsymbol{\theta}_i, \boldsymbol{x}_i)$ and $e_{i,j}$ its j-th component then

$$r_{\mathscr{T}}(\mathscr{C},\mathscr{T}) = (e_{1,1},\ldots,e_{1,n},e_{2,1},\ldots,e_{2,n},\ldots,e_{d_o,n}).$$

The Jacobian of $e_{\mathscr{T}}$ has $(d_{\text{out}}\cdot n)$ rows, one for each component of the residue vector $r_{\mathscr{T}}$, and $(3^{d_{\text{in}}}\cdot d_{\text{out}})$ columns for each component of the control vertices in $\mathscr{C}$.

The second variant does not minimize the components of the error vector $\boldsymbol{e}_i$ separately but its squared length

$$\bar{e}_i = \boldsymbol{e}_i^T\cdot\boldsymbol{e}_i, \quad i = 1,\ldots,n.$$

The resulting residue vector $\bar{r}_{training}$ therefore has only n components.

Symmetry terms

For every control vertex, its neighboring vertices with respect to the i-th degree of freedom are obtained by $\bar{\imath}$ as presented in Eq. (3.17). Let

$$\boldsymbol{a} := \boldsymbol{b}_{(\ldots,\underset{i}{0},\ldots)}, \qquad \boldsymbol{b} := \boldsymbol{b}_{(\ldots,\underset{i}{1},\ldots)}, \qquad \boldsymbol{c} := \boldsymbol{b}_{(\ldots,\underset{i}{2},\ldots)}$$

be such a symmetric triangle. Its isosceles property is expressed by an error term r_Δ called residue. One possibility to define r_Δ is the difference between the squared length of the triangle's legs

$$\boldsymbol{r}_\Delta = \|\boldsymbol{a}-\boldsymbol{b}\|^2 - \|\boldsymbol{b}-\boldsymbol{c}\|^2.$$

This residue is, however, not invariant to the scaling of the triangle such that the choice of the ratio between the legs is more suitable for the residue:

$$\boldsymbol{r}_\Delta = \frac{\|\boldsymbol{a}-\boldsymbol{b}\|^2}{\|\boldsymbol{c}-\boldsymbol{b}\|^2} - 1.$$

The associated partial derivatives with respect to the triangle's vertices are then given by

$$\frac{\partial \boldsymbol{r}_\Delta}{\partial \square} = \begin{cases} \cdot \frac{2\boldsymbol{q}\cdot 0\cdot \boldsymbol{r}^2 - \boldsymbol{q}^2\cdot 2\cdot \boldsymbol{r}\cdot 1}{\boldsymbol{r}^4} & ,\square = \boldsymbol{a}, \\ \cdot \frac{2\boldsymbol{q}\cdot 1\cdot \boldsymbol{r}^2 - \boldsymbol{q}^2\cdot 2\cdot \boldsymbol{r}\cdot (-1)}{\boldsymbol{r}^4} & ,\square = \boldsymbol{b}, \\ \cdot \frac{2\boldsymbol{q}\cdot (-1)\cdot \boldsymbol{r}^2 - \boldsymbol{q}^2\cdot 2\cdot \boldsymbol{r}\cdot 0}{\boldsymbol{r}^4} & ,\square = \boldsymbol{c}. \end{cases}$$

For the regulation of the common angle α, another residue $\boldsymbol{r}_\alpha$ has to be defined. Again, its definition by the ratio of the involved squared distances $\boldsymbol{x}$ (the base of the isosceles triangle) and $\boldsymbol{y}$ (its height, see Figure 4.6)

$$\begin{aligned} \boldsymbol{x} &= {}^1\!/_2 \cdot (\boldsymbol{a}-\boldsymbol{c}) \\ \boldsymbol{y} &= \boldsymbol{b} - {}^1\!/_2 (\boldsymbol{a}+\boldsymbol{c}) \\ \boldsymbol{r}_\alpha &= \frac{\|\boldsymbol{y}\|^2}{\|\boldsymbol{x}\|^2} - \tan^2\alpha \end{aligned}$$

is to be preferred over a non-scale-invariant

$$\boldsymbol{r}_\alpha = \|\boldsymbol{y}\|^2 - \tan^2(\alpha)\cdot\|\boldsymbol{x}\|^2.$$

The partial derivatives for each vertex of the triangle are given by

$$\frac{\partial \boldsymbol{r}_\alpha}{\partial \square} := \begin{cases} \frac{2\boldsymbol{y}\cdot(-1/2)\cdot\boldsymbol{x}^2-\boldsymbol{y}^2\cdot 2\cdot\boldsymbol{x}\cdot 1/2}{\boldsymbol{x}^4} & ,\square=\boldsymbol{a}, \\ \frac{2\boldsymbol{y}\cdot 1\cdot\boldsymbol{x}^2-\boldsymbol{y}^2\cdot 2\cdot\boldsymbol{x}\cdot 0}{\boldsymbol{x}^4} & ,\square=\boldsymbol{b}, \\ \frac{2\boldsymbol{y}\cdot(-1/2)\cdot\boldsymbol{x}^2-\boldsymbol{y}^2\cdot 2\cdot\boldsymbol{x}\cdot(-1/2)}{\boldsymbol{x}^4} & ,\square=\boldsymbol{c}. \end{cases}$$

Algorithm 12: The skeleton algorithm for the steepest descent and the Gauss-Newton methods.

Input: Training data set $\mathscr{D}=\{(\boldsymbol{\theta}_i,\boldsymbol{x}_i): i=1,\ldots,m\}$, Initial control net $\mathscr{C}^0$, maximal number of iterations n, desired accuracy $\varepsilon>0$

Output: Final control net $\mathscr{C}^\star$

$\tilde{\boldsymbol{x}}_i \leftarrow \frac{\boldsymbol{x}_i-\bar{\boldsymbol{x}}}{\max \boldsymbol{x}_{i,j}-\min \boldsymbol{x}_{i,j}}$ // Normalization

$\tilde{\boldsymbol{\theta}}_i \leftarrow \boldsymbol{\theta}_i-\bar{\boldsymbol{\theta}}$ // Normalization

$i \leftarrow 0$

while $\|\boldsymbol{e}(\mathscr{C}^i,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{\theta}})\| > \varepsilon$ *and* $i<n$ **do** // Optimization loop

  Calculate $J(\mathscr{C}^i)$ and $\boldsymbol{e}(\mathscr{C}^i,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{\theta}})$

  $C \leftarrow \begin{cases} J(\mathscr{C}^i)^T\cdot\boldsymbol{e}(\mathscr{C}^i,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{\theta}}), & \text{// steepest descent} \\ (J(\mathscr{C}^i)^T\cdot J(\mathscr{C}^i))^{-1}\cdot J(\mathscr{C}^i)^T\cdot\boldsymbol{e}(\mathscr{C}^i,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{\theta}}), & \text{// Gauss-Newton} \end{cases}$

  $\Delta\boldsymbol{b_i} \leftarrow \left(\iota^{-1}(\boldsymbol{i})\right)$-th column of C

  $\mathscr{C}^{i+1} \leftarrow \{\boldsymbol{b_i}+\nu\cdot\Delta\boldsymbol{b_i} : \boldsymbol{i}\in\mathscr{I}\}$ // Actualization of the control net

  if $\|\boldsymbol{e}(\mathscr{C}^{i+1},\tilde{\boldsymbol{x}},\tilde{\boldsymbol{\theta}})\| < \|\boldsymbol{e}(\mathscr{C}^i,\tilde{\boldsymbol{x}},\tilde{\boldsymbol{\theta}})\|$ **then** // Line search

    $\nu \leftarrow \nu\cdot 2$ // increase stepsize

  else

    $\nu \leftarrow \nu\cdot 1/16$ // reduce step size

  $i \leftarrow i+1$

Algorithm. 11 and Algorithm. 10 show how the partial derivatives for the symmetry terms are computed.

Nonlinear Optimization

The symmetry constraints are formalized by the following nonlinear optimization problem:

$$\mathscr{C}^\star = \arg\min_{\mathscr{C}}\left(e(\mathscr{C},\mathscr{T})\right) = \arg\min_{\mathscr{C}}\left(\boldsymbol{r}(\mathscr{C},\mathscr{T})^T\cdot\boldsymbol{r}(\mathscr{C},\mathscr{T})\right) \tag{4.16}$$

Two numeric optimization techniques were investigated in this work. The first, the *Gauss-Newton* method with line-search for the optimal step size, provides an optimization process that converges quickly to a minimum of the error function. The second is the simpler *steepest descent* that—despite its much slower convergence—is better suited for kinematics with higher numbers of degrees of freedom as it does not rely on memory expensive inversions of the Jacobi matrix. The common structure of the algorithms is shown in Algorithm. 12. They both start with an approximated control net $\mathscr{C}^0$ (preferably the linear approximation computed by Algorithm. 8) which is then refined in every iteration until either a maximal number of itera-

tions (i.e., convergence) is exceeded or a desired accuracy has been reached. The main difference between steepest descent and Gauss-Newton is how the direction, that is, the increment $\Delta \boldsymbol{b_i}$ to the control vertices in $\mathscr{C}^i$ is determined. In the steepest descent variant, the direction is computed from the product of the Jacobian and the residues vector

$$C = J(\mathscr{C}^i)^T \cdot \boldsymbol{e}(\mathscr{C}^i, \tilde{\boldsymbol{x}}, \tilde{\boldsymbol{\theta}}), \tag{4.17}$$

where $\Delta \boldsymbol{b_i}$ is the $\left(\iota^{-1}(\boldsymbol{i})\right)$-th column of the resulting matrix C. That way, it has a high resemblance to the δ-rule presented in Section 4.3 which offers the most basic incremental learning of the KBM model. As both are steepest descents, Eq. (4.17) can be considered the batch learning extension of the δ-rule when omitting the nonlinear error terms.

Gauss-Newton, however, creates the first taylor expansion of the error function in each step and determines its root. It then moves the control parameters into the direction of this root:

$$C = (J(\mathscr{C}^i)^T \cdot J(\mathscr{C}^i))^{-1} \cdot J(\mathscr{C}^i)^T \cdot \boldsymbol{e}(\mathscr{C}^i, \mathscr{T}) \tag{4.18}$$

Again, the increment $\Delta \boldsymbol{b_i}$ lies in the $\left(\iota^{-1}(\boldsymbol{i})\right)$-th column of C. Then, control net is updated according to

$$\mathscr{C}^{i+1} \leftarrow \{\boldsymbol{b_i} + \nu \cdot \Delta \boldsymbol{b_i} : \boldsymbol{i} \in \mathscr{I}\} \tag{4.19}$$

which features a step size factor ν that has to be chosen in each step by a line search. This means that ν has to be increased every time the error reduces (the norm of the residue vector) and decreased again if the error increases. In the experiments later, ν is doubled or divided by the value 16 which led to good results.

Both algorithms share the same basic structure and while Gauss-Newton converges in fewer iterations, steepest descent is the better choice for six degrees of freedom or more because it avoids the inversion of possibly huge matrices that become too expensive to compute.

A problem common to both algorithms is that the errors of the approximation depend on the magnitude of the Cartesian positions $\boldsymbol{x}_i$ while the symmetry constraints are invariant. If the arm is ten times longer (e.g., when using different units), for instance, the squared error of the approximation becomes bigger and the nonlinear terms are likely to be neglected. Therefore, the Cartesian coordinates of the training data should be normalized such that every coordinate lies in between ± 1:

$$\tilde{\boldsymbol{x}_i} = \frac{\boldsymbol{x}_i - \bar{\boldsymbol{x}}}{\max \boldsymbol{x}_{i,j} - \min \boldsymbol{x}_{i,j}}, \quad \forall \boldsymbol{x}_i. \tag{4.20}$$

Similarly, the input variables should be subtracted by their mean values:

$$\tilde{\boldsymbol{\theta}}_i = \boldsymbol{\theta}_i - \bar{\boldsymbol{\theta}}, \quad \forall \boldsymbol{\theta}_i \tag{4.21}$$

Further, the error function is a weighted sum and the weights play an important role in the optimization. An initial good guess for tuning these parameters, is the inverse value of the number of residues in the residue vector, that is,

$$\omega_1 = \frac{1}{3^{d_{\text{in}}} - 2^{d_{\text{in}}}} = \omega_2, \text{ and } \omega_3 = \frac{1}{d_{\text{out}} \cdot n}, \tag{4.22}$$

where n is the number of training samples. Depending on the expected noise in the data, ω_3 should be adjusted differently.

4.4.3. Summary

The nonlinear optimization presented in this section is by far more complex than the linear optimizations presented earlier. Further, the computational complexity is greater and determination of the model slower. On the other hand, the maintenance of the symmetry makes the learning less prone to overfitting and its construction offers an interesting insight into the structure of the learned model. Its field of application therefore is especially recommended if the configuration space of the robot cannot sufficiently be explored and no a-priori model of the robot is available for initialization.

5. Dimensionality reduction for sensorimotor learning

5.1. Introduction

The application of machine learning for creating sensorimotor maps provides a reliable and versatile method to foster robot control and planning. Especially the previously presented KBM offer a suitable model specialized to represent the nonlinear relations between the input and output data while remaining a sufficient degree of generality. The linearization of the problem unveils a problem inherent to sensorimotor learning in general. The more independent and sequential degrees of freedom a kinematics consists of, the more training samples are required to deduce a valid model. As this number grows exponentially, this problem is often referred to as 'the curse of dimensionality' (Bellman, 1960). Local non-parametric learning such as LWPR and XSCF (see Section 2.3.4) avoid this problem by modeling the maps by a growing composition of linear models. While this leads to quick results when learning models that are valid in a vicinity of the observed samples, covering a large workspace requires a larger amount of samples compared to the KBM.

This chapter covers a novel method that directly addresses the curse of dimensionality—at least so in the case of learning forward kinematics models. It is based on the decomposition techniques shown in Section 2.3.6 but provides significant improvements. The fundamental idea underlying the decomposition is the division of the complex learning problem into a set of smaller problems that can be handled with ease. A kinematics is divided into smaller chains with respect to the number of active degrees of freedom which significantly reduces the number of required samples due to the exponentiality of the problem. For instance, the decomposition of a kinematics with an even number of sequential degrees of freedom requires $\mathcal{O}(e^{2 \cdot d_{\text{in}}})$ training samples to be learned. A decomposition into two evenly long *sub-chains* (also called *partial kinematics* in the following paragraphs) requires a number of only $\mathcal{O}(e^{d_{\text{in}}})$ which effectively means a reduction to its square root.

Decompositions exploit the structure of the forward kinematics which can be expressed as a series of rigid body transformations. Consequently, they are limited to this type of functions only and reduce the generality of the machine learning. In addition, there rises a number of drawbacks: *i.)* The input variables have to be sorted according to their *lateral distance*—the number of active joints that separate the respective joint from the robot's base. This order is required to divide the input data into *medial* close to the robot's base and *lateral joint angles* close to the end-effector. *ii.)* The output must be rigid transformations out of the special Eucledean group SO_3, homogenous transformation matrices for instance, and the sensors must be able to track the end-effector in spatial 6D coordinates. *iii.)* A more complicated sensor setup may be required (Ruiz de Angulo and Torras, 2008) to track multiple markers on the robot's body. Especially when learning from self-observation on a humanoid robot, this constraint is a hindrance as it limits the robot's movability

when all markers have to be in the field of vision at the same time. *iv.)* The strong limitation to special types of robots (Ruiz de Angulo and Torras, 2005b), e.g., when the last three joint axes intersect in a single joint.

In the following, a new decomposition scheme is presented that resolves the last two problems at the expense of a more constrained exploration strategy. It is well suited for learning from self-observation and, despite the complexity of the underlying affine geometric calculus, it is a very intuitive and minimalistic approach that can be implemented with minimal effort on a robot. It has been presented in (Ulbrich et al., 2012b).

5.2. Kinematics Decomposition

The basic idea behind this decomposition is the reduction of the learning problem by decomposing the kinematics into subchains that can be learned more easily. Therefore, the joint angles, on which the kinematics depends

$$\boldsymbol{\theta} = \left(\theta_1, \ldots, \theta_{d_{\text{in}}}\right),$$

are partitioned into two sets of adjacent joint angles, for instance,

$$\boldsymbol{\xi} = \left(\xi_1, \ldots, \xi_k\right), \quad \xi_i = \theta_i, i \leq k$$

and

$$\boldsymbol{\mu} = \left(\mu_1, \ldots, \mu_{d_{\text{in}}-k}\right), \quad \mu_i = \theta_{i+k}, i \leq d_{\text{in}} - k.$$

The *medial joint angles* are then contained in the vector $\boldsymbol{\xi}$ and the *lateral* in $\boldsymbol{\mu}$. The kinematics can be separated into more parts depending on even fewer input variables. It is beneficial to create parts that depend on an equal number of inputs, such that k equals $d_{\text{in}}/2$ as the effect of the decomposition is then maximal. The forward kinematics

$$k : \mathbb{R}^{d_{\text{in}}} \rightarrow SO_3 \tag{5.1}$$

can then be written as the product of two shorter kinematic functions $k_\mu(\boldsymbol{\mu})$ and $k_\xi(\boldsymbol{\xi})$ which will be used as helper functions

$$\begin{aligned} k(\boldsymbol{\theta}) &= M_{0,1}(\theta_1) \cdot \ldots \cdot M_{d_{\text{in}}-1,d_{\text{in}}}(\theta_{d_{\text{in}}}), \\ &= \underbrace{M_{0,1}(\xi_1) \cdot \ldots \cdot M_{k-1,k}(\xi_k)}_{k_\xi} \cdot \underbrace{M_{k,k+1}(\mu_1) \cdot \ldots \cdot M_{d_{\text{in}}-1,di}(\mu_{n-k})}_{k_\mu}, \\ &= k_\xi(\boldsymbol{\xi}) \cdot k_\mu(\boldsymbol{\mu}) \end{aligned} \tag{5.2}$$

where the $M_\square \in SO_3$ are rigid transformations (e.g., homogeneous matrices) between adjacent joints. Therefore, the joint angles in $\boldsymbol{\theta}$ have sorted by their lateral distance (the number of joints separating the from the kinematic's base). For convenience, $k(\boldsymbol{\theta}) = k(\boldsymbol{\xi}, \boldsymbol{\mu})$ is written to enforce the distinction of the two sets of joint angles.

Now that the definitions necessary to describe the decompositions are available, the first function in the decompositions can be defined as

$$k_1(\boldsymbol{\xi}) := k(\boldsymbol{\xi}, \tilde{\boldsymbol{\mu}}) = k_\xi(\boldsymbol{\xi}) \cdot k_\mu(\tilde{\boldsymbol{\mu}}) = k_\xi(\boldsymbol{\xi}) \cdot C_{\tilde{\boldsymbol{\mu}}} \tag{5.3}$$

for an arbitrary but fixed medial joint configuration $\tilde{\boldsymbol{\mu}}$ where $C_{\tilde{\boldsymbol{\mu}}} = k_\mu(\tilde{\boldsymbol{\mu}})$ is a constant transformation associated to the fixed joint values $\tilde{\boldsymbol{\mu}}$. The second function in the decomposition which depends on the lateral angles $\boldsymbol{\mu}$ is defined by the relation between k and k_1 such that

$$k(\boldsymbol{\xi}, \boldsymbol{\mu}) = k_1(\boldsymbol{\xi}) \cdot k_2(\boldsymbol{\mu}) \tag{5.4}$$

holds. Solving this equation for k_2 leads to

$$\begin{aligned} k_2(\boldsymbol{\mu}) &:= (k_{\boldsymbol{\xi}}(\boldsymbol{\xi}) \cdot k_{\boldsymbol{\mu}}(\tilde{\boldsymbol{\mu}}))^{-1} \cdot k_{\boldsymbol{\xi}}(\boldsymbol{\xi}) \cdot k_{\boldsymbol{\mu}}(\boldsymbol{\mu}) \\ &= k_{\boldsymbol{\mu}}(\tilde{\boldsymbol{\mu}})^{-1} \cdot k_{\boldsymbol{\xi}}(\boldsymbol{\xi})^{-1} \cdot k_{\boldsymbol{\xi}}(\boldsymbol{\xi}) \cdot k_{\boldsymbol{\mu}}(\boldsymbol{\mu}) \\ &= C_{\tilde{\boldsymbol{\mu}}}^{-1} \cdot \ k_{\boldsymbol{\mu}}(\boldsymbol{\mu}). \end{aligned} \tag{5.5}$$

which shows that k_2 also represents a valid kinematic function that depends only on the $n-k$ lateral joint angles in $\boldsymbol{\mu}$. The product of this two functions k_1 and k_2 that depend on two disjoint sub sets of joint angles is equivalent to the real forward kinematics.

Note that, alternatively, the decomposition can reversely be defined by using a fixed set of medial angles $\tilde{\boldsymbol{\xi}}$ and the kinematic functions

$$\bar{k}_1(\boldsymbol{\mu}) = k(\tilde{\boldsymbol{\xi}}, \boldsymbol{\mu}) \quad \text{and} \quad \bar{k}_2(\boldsymbol{\xi}) = k(\boldsymbol{\xi}, \boldsymbol{\mu}) \cdot \bar{k}_1(\boldsymbol{\mu})^{-1}.$$

The kinematics composition is analogously obtained from the definition of $\bar{k}_2$,

$$k(\boldsymbol{\xi}, \boldsymbol{\mu}) = \bar{k}_2(\boldsymbol{\xi}) \cdot \bar{k}_1(\boldsymbol{\mu})$$

If a decomposition into more than two sub chains is required, then the same procedure as described above can be applied to either k_1 or k_2 which is possible as both represent valid kinematics. That way any number of subchains can be produced recursively for a desired length of d_{in}/d for each chain. The recursion starts with one single subdivision into two chains of length d_{in}/d and $(d_{\text{in}} - d_{\text{in}}/d)$. In the following recursions, the larger chain is divided into two additional chains such that one subchain is again of length d_{in}/d. The algorithm terminates when all chains are of length d_{in}/d or smaller. Algorithm. 13 shows the pseudocode for the algorithm. For the remainder of this chapter, however, we assume a single decomposition without loss of generalization.

Algorithm 13: Algorithm for the recursive definition of the decomposition.

Input: Forward kinematics $k = M_{0,1}(\theta_1) \cdot \ldots \cdot M_{d_{\text{in}}-1,d_{\text{in}}}(\theta_{d_{\text{in}}}) \in SO_3$, length of partial kinematics $l = {}^{d_{\text{in}}}/_{n}$, reference configuration $\tilde{\theta}$

Output: Decomposition $k_1, \ldots, k_n$

Function *RecursiveDecomposition*($k(\boldsymbol{\theta})$),l,$\tilde{\boldsymbol{\theta}}$)

    // Obtain the first decomposition

    $\boldsymbol{\xi} := \left(\theta_1, \ldots, \theta_l\right)^T$

    $\boldsymbol{\mu} := \left(\theta_{l+1}, \ldots, \theta_{d_{\text{in}}}\right)^T$

    $\tilde{\boldsymbol{\mu}} := \left(\tilde{\theta}_{l+1}, \ldots, \tilde{\theta}_{d_{\text{in}}}\right)^T$

    $k_1(\boldsymbol{\xi}) := k_\xi(\boldsymbol{\xi}) \cdot k_\mu(\tilde{\boldsymbol{\mu}}) = k_\xi(\boldsymbol{\xi}) \cdot C_{\tilde{\boldsymbol{\mu}}}$

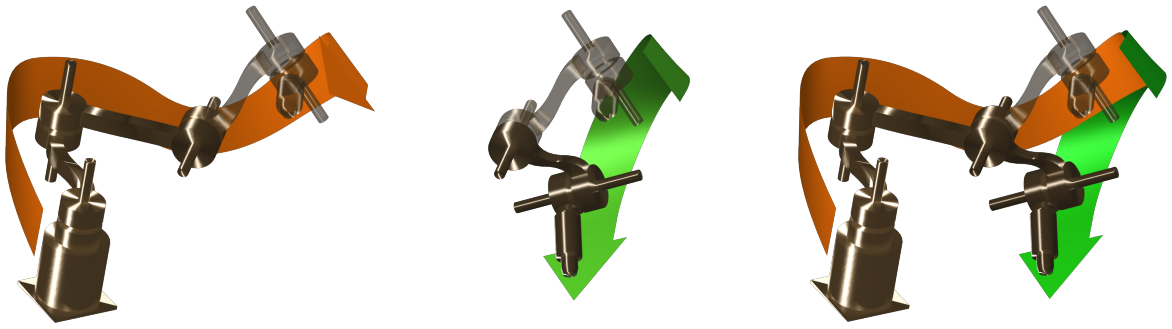
    if $d_{in} \leq 2 \cdot l$ **then**

        $k_2 := C_{\tilde{\boldsymbol{\mu}}}^{-1} \cdot k_\mu(\boldsymbol{\mu})$

        return *list*(k_1,k_2)

    else

        return *list*(k_1, *RecursiveDecomposition*($M_{l,l+1}(\theta_1) \cdot \ldots \cdot M_{d_{in}-1,d_{in}}$, l,$(\tilde{\theta}_{l+1}, \ldots, \tilde{\theta}_{d_{in}})^T$)



(a) $k_1(\boldsymbol{\xi}) = k_\xi(\boldsymbol{\xi}) \cdot C_{\tilde{\boldsymbol{\mu}}}$ (b) $k_2(\boldsymbol{\mu}) = C_{\tilde{\boldsymbol{\mu}}}^{-1} \cdot k_\mu(\boldsymbol{\mu})$ (c) $k(\boldsymbol{\xi},\boldsymbol{\mu}) = k_1(\boldsymbol{\xi}) \cdot k_2(\boldsymbol{\mu})$

Figure 5.1.: Illustration of the kinematic decomposition without intermediate markers. The decomposition of the forward kinematics k into the two chains k_1 (a) and k_2 (b)is explained on the example of fictive robot with four revolute joints. The partial kinematics k_1 is the transformation from the robot base to the end-effector while the most lateral joints are fixed. In (a) the stationary part of the robot is rendered transparently. The second partial kinematics k_2 in (a) combines the stationary part from the previous image with the kinematics of the lateral joints. Consequently, it is the transformation from the tail of k_1 to the real end-effector pose of k. When k_1 and k_2 the resulting kinematics equal the real kinematics k. *Source: (Ulbrich et al., 2012b)*

5.3. Kinematics Composition

The forward kinematics can be assembled as shown in Eq. (5.2). For learning the whole kinematics, learning systems (this could be neural networks or the KBM presented earlier) are applied to learn each of the of the decomposition such that

$$n(\boldsymbol{\xi}, \boldsymbol{\mu}) = n_1(\boldsymbol{\xi}) \cdot n_2(\boldsymbol{\mu}). \tag{5.6}$$

This provides the justification that k—the number of joints in each chain—should be chosen close to $d_{\text{in}}/2$ as, in general, the number of samples required for training the systems $n_1(\boldsymbol{\xi})$ and $n_2(\boldsymbol{\mu})$ depends on the number of joints in the partitioned joint vector $\boldsymbol{\xi}$ and $\boldsymbol{\mu}$. The total number required for learning is then minimal if k equals $d_{\text{in}}/2$ and exponentially grows with the distance to this value.

For the inverse kinematics, the joint coordinates $\boldsymbol{\theta}$ form a solution to the inverse kinematics if and only if, given a target pose $X \in SE_3$ holds

$$k(\boldsymbol{\xi}, \boldsymbol{\mu}) = k_1(\boldsymbol{\xi}) \cdot k_2(\boldsymbol{\mu}) = X, \tag{5.7}$$

which can be approximated with the learning systems by

$$n(\boldsymbol{\xi}, \boldsymbol{\mu}) = n_1(\boldsymbol{\xi}) \cdot n_2(\boldsymbol{\mu}) = X. \tag{5.8}$$

using Eq. (5.3) and Eq. (5.5), this can be transformed to

$$\begin{aligned} & k_{\boldsymbol{\xi}}(\boldsymbol{\xi}) \cdot C_{\tilde{\boldsymbol{\mu}}} \cdot C_{\tilde{\boldsymbol{\mu}}}^{-1} \cdot k_{\boldsymbol{\mu}}(\boldsymbol{\mu}) &&= X \\ \Leftrightarrow \quad & k_{\boldsymbol{\xi}}(\boldsymbol{\xi}) \cdot k_{\boldsymbol{\mu}}(\boldsymbol{\mu}) &&= X \\ \Leftrightarrow \quad & k_{\boldsymbol{\xi}}(\boldsymbol{\xi}) &&= X \cdot k_{\boldsymbol{\mu}}(\boldsymbol{\mu})^{-1}. \end{aligned} \tag{5.9}$$

This equality is very similar to the decomposition proposed in (Ruiz de Angulo and Torras, 2008). As a reminder, this decomposition can be better described by kinematic functions k_ξ and k_μ in Eq. (5.2) and requires at least two (6D) markers on the arm for learning—on the end-effector and on the elbow for instance. The decomposition presented in this section requires only the marker on the end-effector. This important property will be highlighted in the following paragraph.

Most numerical methods for solving the inverse kinematics are based on the relation between the differences in the joint angles and the resulting motion of the end-effector

$$J(\boldsymbol{\theta}) \cdot \Delta\boldsymbol{\theta} \approx \Delta\boldsymbol{x} \quad \Longleftrightarrow \quad \Delta\boldsymbol{\theta} = J(\boldsymbol{\theta})^{+} \cdot \Delta\boldsymbol{x},$$

where $J(\boldsymbol{\theta})$ denotes the Jacobian matrix given a joint configuration $\boldsymbol{\theta}$ and $J(\boldsymbol{\theta})^{+}$ its *Moore-Penrose* or *pseudo* inverse (Whitney, 1969; Liegeois, 1977; Li et al., 2001; Xia et al., 2005). This relation is then used to iteratively find a solution for a given target position or pose $\boldsymbol{x}$. The decomposition k_1 and k_2 allows the approximation of the Jacobian by means of the involved learning systems n_1 and n_2 after learning. The partial derivatives which are in the columns of this matrix are easily obtained by combining the networks'

outputs and their partial derivatives

$$\frac{\partial n(\boldsymbol{\xi},\boldsymbol{\mu})}{\partial \xi_i} = \frac{\partial n_1(\boldsymbol{\xi})}{\partial \xi_i} \cdot n_2(\boldsymbol{\mu}) \quad \text{and} \quad \frac{\partial n(\boldsymbol{\xi},\boldsymbol{\mu})}{\partial \mu_j} = n_1(\boldsymbol{\xi}) \cdot \frac{\partial n_2(\boldsymbol{\mu})}{\partial \mu_j}. \tag{5.10}$$

If the partial derivatives cannot be accessed through the learning systems, the difference quotient can be used as an approximation, for instance,

$$\frac{\partial n(\boldsymbol{\xi},\boldsymbol{\mu})}{\partial \xi_i} = \frac{n_1(\boldsymbol{\xi}) - n_1(\boldsymbol{\xi}+\Delta\boldsymbol{\xi})}{\|\Delta\boldsymbol{\xi}\|} \cdot n_2(\boldsymbol{\mu})$$

for small values of $\Delta\boldsymbol{\mu}$.

5.4. Learning

The decomposition of a kinematics $k(\boldsymbol{\xi},\boldsymbol{\mu})$ into $k_1(\boldsymbol{\xi})$ and $k_2(\boldsymbol{\mu})$ results in a significant reduction of required training samples. On the downside, the learning procedure is generally more complex. This paragraph suggests two different algorithms that describe how training samples $(\boldsymbol{\theta}_i, X_i)$ have to be processed to enable the learning of partial kinematics with two learning systems n_1 and n_2 that can be any general function approximation method or a specialized model like the KBM presented earlier. The data acquisition requires the end-effector to be localizable in 6D coordinates, that is, that $X_i \in SO_3$ encoded as homogenous matrices for instance. The localization of additional body parts (e.g., the elbow link) is not required in this approach which facilitates the application on a humanoid robot that learns from pure self-observation with stereo cameras build into the robot's head.

5.4.1. Independent learning

Learning each partial kinematics $k_1(\boldsymbol{\xi})$ and $k_2(\boldsymbol{\mu})$ independently is the genuine approach that can directly be deduced from Eq. (5.3) and Eq. (5.5). The approach is separated into two steps shown inAlgorithm. 14 and Algorithm. 15 which have to be executed in sequential order. In the first step, training data has to be generated such that the medial joints in the first training set $\boldsymbol{\xi}_i$ vary

$$\boldsymbol{\xi}_i \neq \boldsymbol{\xi}_j,$$

(e.g., are generated from random movements) and the lateral joint positions $\boldsymbol{\mu}_i$ remain at a reference configuration $\tilde{\boldsymbol{\mu}}$. Then the training data for n_1 has the form $(\boldsymbol{\xi}_i, X_i)$. The second step can theoretically operate on completely arbitrary joint configurations $(\boldsymbol{\theta}_i, X_i)$

$$\begin{aligned} &\boldsymbol{\theta}_i \neq \boldsymbol{\theta}_j, \quad i \neq j \\ &X_i = K(\boldsymbol{\xi}_i, \tilde{\boldsymbol{\mu}})^{-1} \cdot K(\boldsymbol{\xi}_i, \boldsymbol{\mu}_i), \end{aligned}$$

Algorithm 14: Learning of $k_1(\boldsymbol{\xi})$.

Input: Training data $\mathscr{T} = \{(\boldsymbol{\xi}_i, \tilde{\boldsymbol{\mu}}, X_i), i = 1, \ldots, n\}$
Output: Learning system $n_1(\boldsymbol{\xi})$ that approximates $k_1(\boldsymbol{\xi})$

1 **foreach** $(\boldsymbol{\xi}_i, \tilde{\boldsymbol{\mu}}, X_i) \in \mathscr{T}$ **do**
2 Move to $(\boldsymbol{\xi}_i, \tilde{\boldsymbol{\mu}})$ and observe $X_i = k(\boldsymbol{\xi}_i, \tilde{\boldsymbol{\mu}})$
3 Learn n_1 with $\boldsymbol{\xi}_i$ as input and X_i as output.

Algorithm 15: Learning of $k_2(\boldsymbol{\mu})$.

Input: Training data $\mathscr{T} = \{(\tilde{\boldsymbol{\xi}}, \boldsymbol{\mu}_i, X_i), i = 1, \ldots, n\}$
Output: Learning system $n_2(\boldsymbol{\mu})$ that approximates $k_2(\boldsymbol{\mu})$

1 Move to $(\tilde{\boldsymbol{\xi}}, \tilde{\boldsymbol{\mu}})$ and observe $C_{\tilde{\boldsymbol{\mu}}} = K(\tilde{\boldsymbol{\xi}}, \tilde{\boldsymbol{\mu}})^{-1}$
2 **foreach** $(\tilde{\boldsymbol{\xi}}, \boldsymbol{\mu}_i, X_i) \in \mathscr{T}$ **do**
3 Move to $(\tilde{\boldsymbol{\xi}}, \boldsymbol{\mu}_i)$ and observe $X_i = k(\hat{\boldsymbol{\zeta}}, \boldsymbol{\mu}_i)$
4 Learn n_2 with $\boldsymbol{\mu}_i$ as input and $C_{\tilde{\boldsymbol{\mu}}} \cdot X_i$ as output.

still resulting in training data $(\boldsymbol{\mu}_i, X_i)$. However, if $\boldsymbol{\xi}_i$ remains at fixed reference position $\tilde{\boldsymbol{\xi}}$, the transformation $K(\boldsymbol{\zeta}_i, \tilde{\boldsymbol{\mu}})^{-1}$ is constant and can be obtained already prior to the second step thus saving one movement in each iteration. In summary, both steps boil down to basically varying one of the partial kinematics while the other remains constant.

Especially Algorithm. 15 can be modified to match local conditions. For instance, it can be executed with different values for $\tilde{\boldsymbol{\xi}}$ if $\boldsymbol{\mu}$ is constrained for some values of $\boldsymbol{\xi}$ (e.g., to keep the end-effector in the field of view). Even if it is required to use a different $\boldsymbol{\xi}_i$ for each $\boldsymbol{\mu}_i$, training n_2 with only one movement per iteration is possible. This requires the inclusion of the first learning system's output $n_1(\boldsymbol{\xi}_i)$ for the generation of the training data

$$X_i = n_1(\boldsymbol{\xi}_i)^{-1} \cdot K(\boldsymbol{\xi}_i, \boldsymbol{\mu}_i),$$

derived from Eq. (5.8). On the downside, the data depends on an approximation of k_1 which is an additional source of error. Therefore, k_1 should have a low input dimension and sufficiently training be provided such that the error becomes negligible.

5.4.2. Concurrent learning

Independent learning as presented above requires pure *offline learning due* to its sequential processing and restrictive joint movements. In order to integrate learning into the normal working operation (i.e., *online learning*), a concurrent learning of the learning systems $n_1(\boldsymbol{\xi})$ and $n_2(\boldsymbol{\mu})$ must be achieved. The strategy presented in the following is a parallelization of the learning of the partial kinematics presented in Algorithm. 16. Interestingly, it permits refinement of decompositions previously learned by independent learning from *arbitrary* joint configurations and joint movements.

The key to concurrent learning lies in Eq. (5.8). The partial kinematics can be expressed in dependence of

each other

$$k_1(\boldsymbol{\xi}_i) = k(\boldsymbol{\xi}_i, \boldsymbol{\mu}_i) \cdot k_2^{-1}(\boldsymbol{\mu}_i) \tag{5.11}$$
$$k_2(\boldsymbol{\xi}_i) = k^{-1}(\boldsymbol{\xi}_i) \cdot k(\boldsymbol{\xi}_i, \boldsymbol{\mu}_i) \tag{5.12}$$

which means that the output values for n_1 and n_2 are given implicitly by

$$n_1(\boldsymbol{\xi}_i) = X_i \cdot n_2(\boldsymbol{\mu}_i)^{-1}, \tag{5.13}$$
$$n_2(\boldsymbol{\mu}_i) = n_1(\boldsymbol{\xi}_i)^{-1} \cdot X_i \tag{5.14}$$

The learned partial kinematics

Note that the reference joint configuration $\tilde{\boldsymbol{\mu}}$ is completely missing in this definition and, in fact, learning as presented in Algorithm. 16 tends to converge to functions that do satisfy

$$k(\boldsymbol{\xi}_i, \boldsymbol{\mu}_i) = n_1(\boldsymbol{\xi}_i) \cdot n_2(\boldsymbol{\mu}_i),$$

but do not directly approximate k_1 and k_2 anymore. The functions they represent, however, are of the following form

$$n_1(\boldsymbol{\xi}) = k_1(\boldsymbol{\xi}) \cdot C^{-1} \qquad\qquad n_2(\boldsymbol{\mu}) = C \cdot k_2(\boldsymbol{\mu}), \tag{5.15}$$

which implies that

$$n_1(\boldsymbol{\xi}) \cdot n_2(\boldsymbol{\mu}) = k(\boldsymbol{\xi}, \boldsymbol{\mu})$$

for a constant transformation C. Because of

$$n_2(\tilde{\boldsymbol{\mu}}) = C \cdot k_\mu^{-1}(\tilde{\boldsymbol{\mu}}) \cdot k_\mu(\tilde{\boldsymbol{\mu}})$$

(see Eq. (5.5)), C can be identified as $n_2(\tilde{\boldsymbol{\mu}})$. To prove that these functions approximated by n_1 and n_2 are unique, first two functions ε_1 and ε_2 are defined such that

$$\varepsilon_1(\boldsymbol{\xi}) = k_1(\boldsymbol{\xi})^{-1} \cdot n_1(\boldsymbol{\xi}) \tag{5.16}$$
$$\varepsilon_2(\boldsymbol{\mu}) = n_2(\boldsymbol{\mu}) \cdot k_2(\boldsymbol{\mu})^{-1}. \tag{5.17}$$

The product of both functions

$$\varepsilon_1(\boldsymbol{\xi}) \cdot \varepsilon_2(\boldsymbol{\mu}) = k_1(\boldsymbol{\xi})^{-1} \cdot \underbrace{n_1(\boldsymbol{\xi}) \cdot n_2(\boldsymbol{\mu})}_{k(\boldsymbol{\xi}, \boldsymbol{\mu})} \cdot k_2(\boldsymbol{\mu})^{-1}, \tag{5.18}$$
$$= k_1(\boldsymbol{\xi})^{-1} \cdot \underbrace{k(\boldsymbol{\xi}, \boldsymbol{\mu}) \cdot k_2(\boldsymbol{\mu})^{-1}}_{k_1(\boldsymbol{\xi})} = I \tag{5.19}$$

is the identity matrix I. Because ε_1 and ε_2 depend on two disjoint sets of variables, their dependencies on their input variables cannot be canceled out by multiplication. The only possible solution for the product to be the identity matrix consequently is that ε_1 and ε_2 suffice

$$\varepsilon_1(\boldsymbol{\xi}) = \bar{C}^{-1}, \qquad \varepsilon_2(\boldsymbol{\mu}) = \bar{C},$$

that is, are constant transformations. The substitution of these values into Eq. (5.16) and applying Eq. (5.15) yields

$$\begin{aligned} n_2^{-1}(\tilde{\boldsymbol{\mu}}) &= k_1(\boldsymbol{\xi})^{-1} \cdot n_1(\boldsymbol{\xi}) \\ n_2(\tilde{\boldsymbol{\mu}}) &= n_2(\boldsymbol{\mu}) \cdot k_2(\boldsymbol{\mu})^{-1}, \\ \bar{C} &= C = n_2(\tilde{\boldsymbol{\mu}}) \end{aligned}$$

which shows that Eq. (5.15) is the only possible form of the equations represented by n_1 and n_2. □

The learned partial kinematics n_1 and n_2 consequently approximate a valid decomposition, but it should be noted that they may change significantly as soon as switching from independent to concurrent learning.

Deformations learnable with only one function

From the multitude of functions that are valid decompositions rises another potential advantage. Applying online refinement to only one learning system n_1 or n_2 at a time may be sufficient to adapt to certain kinematics changes which is significantly faster than refining both systems at the same time. The most prominent case is holding a tool such that the kinematics is modified to

$$\bar{k}(\boldsymbol{\xi}, \boldsymbol{\mu}) = k(\boldsymbol{\xi}, \boldsymbol{\mu}) \cdot T,$$

where $T \in SO_3$ is the rigid transformation to the tool tip. The quickest way to react to this modification is to refine only n_2.

In general, when only n_1 actively learns in online application, it can adapt to a modified kinematics $\bar{k}$ with partial kinematics $\bar{k}_1$ and $\bar{k}_2$ if there exists a constant transformation C that satisfies the equation

$$\bar{k}(\boldsymbol{\xi}, \boldsymbol{\mu}) \cdot n_2(\boldsymbol{\mu})^{-1} = \bar{k}_1(\boldsymbol{\xi}) \cdot C. \tag{5.20}$$

On the left-hand side, is the function learned by n_1 (see Eq. (5.11)) and, on the right-hand side, the form all possible functions that n_1 can encode must suffice (see Eq. (5.15)). A simpler condition can be found if n_2 can be assumed have already been learned before, that is, there exists a constant transformation D with

$$n_2(\boldsymbol{\mu}) = D \cdot k_2(\boldsymbol{\mu}).$$

Then, the kinematics is learnable if and only if there exists the same constant transformation C with

$$\bar{k}_2(\boldsymbol{\mu}) = C \cdot k_2(\boldsymbol{\mu}). \tag{5.21}$$

This relation implies Eq. (5.20)

$$\begin{aligned}
\bar{k}(\boldsymbol{\xi},\boldsymbol{\mu}) \cdot n_2(\boldsymbol{\mu})^{-1} &= \bar{k}(\boldsymbol{\xi},\boldsymbol{\mu}) \cdot (D \cdot k_2(\boldsymbol{\mu}))^{-1} \\
&= \bar{k}(\boldsymbol{\xi},\boldsymbol{\mu}) \cdot k_2(\boldsymbol{\mu})^{-1} \cdot D^{-1} \\
&= \bar{k}(\boldsymbol{\xi},\boldsymbol{\mu}) \cdot (C^{-1} \cdot \bar{k}_2(\boldsymbol{\mu}))^{-1} \cdot D^{-1} \\
&= \bar{k}(\boldsymbol{\xi},\boldsymbol{\mu}) \cdot \bar{k}_2(\boldsymbol{\mu})^{-1} \cdot C \cdot D^{-1} \\
&= \bar{k}_1(\boldsymbol{\xi}) \cdot C \cdot D^{-1}.
\end{aligned}$$

The deformations that can be absorbed by learning only the refinement of n_2 can be analogously identified if there exists a constant transformation C such that

$$\bar{k}_1(\boldsymbol{\xi}) = k_1(\boldsymbol{\xi}) \cdot C. \tag{5.22}$$

which covers the case of tool use described in Eq. (5.4.2) but also erroneous calibrations of the camera leading to a scaling of the sensor data. Learning only n_1 online is not possible because the condition Eq. (5.21)

$$\begin{aligned}
\bar{k}_2(\boldsymbol{\mu}) &= \bar{k}(\boldsymbol{\xi},\boldsymbol{\mu})^{-1} \cdot k'(\boldsymbol{\xi},\tilde{\boldsymbol{\mu}}) \\
&= (k(\boldsymbol{\xi},\tilde{\boldsymbol{\mu}}) \cdot C)^{-1} \cdot K(\boldsymbol{\xi},\boldsymbol{\mu}) \cdot C \\
&= C^{-1} \cdot k(\boldsymbol{\xi},\tilde{\boldsymbol{\mu}})^{-1} \cdot k(\boldsymbol{\xi},\boldsymbol{\mu}) \cdot C \\
&= C^{-1} \cdot k_2(\boldsymbol{\mu}) \cdot C
\end{aligned}$$

holds for $C = I$ only.

Distribution of the error

Learning n_1 and n_2 is highly interdependent. In each iteration, the corrections to their internal representation aims at the reduction of the same residue

$$\boldsymbol{r}_i = \| n_1(\boldsymbol{\xi}_i) \cdot n_2(\boldsymbol{\mu}_i) - X_i \|$$

which has to be computed *before* any of the system is modified. The most crucial part of concurrent learning is how much of the error in the residue is to be absorbed by each n_1 and n_2. If both systems try to absorb the same complete error at the same time, the prediction error $n_1(\boldsymbol{\xi}_i) \cdot n_2(\boldsymbol{\mu}_i) - X_i$ will just have the opposite sign and learning has no effect. Consequently, a learn ratio $\nu \in (0,1)$ has to be defined such that two different

Algorithm 16: Simultaneous learning of $K_1(\boldsymbol{\xi})$ and $K_2(\boldsymbol{\mu})$.

Input: Training data $\mathscr{T} = \{(\boldsymbol{\xi}_i, \tilde{\boldsymbol{\mu}}, X_i), i = 1, \ldots, n\}$, initialized learning systens n_1 and n_2, learn rate ν
Output: Refined learning systems n_1 and n_2

foreach $(\boldsymbol{\xi}_i, \boldsymbol{\mu}_i, X_i) \in \mathscr{T}$ **do**
    Move to $(\boldsymbol{\xi}_i, \boldsymbol{\mu}_i)$ and observe $X_i = k(\boldsymbol{\xi}_i, \boldsymbol{\mu}_i)$
    Set $X_{i,1} := X_i \cdot n_2(\boldsymbol{\mu}_i)^{-1}$ and $X_{i,2} := n_1(\boldsymbol{\xi}_i)^{-1} \cdot T_i$
    Learn n_1 with $\boldsymbol{\xi}_i$ as input and error signal $\nu \cdot (n_1(\boldsymbol{\xi}_i) - X_{i,1})$ as output.
    Learn n_2 with $\boldsymbol{\mu}_i$ as input and error signal $(1-\nu) \cdot (n_2(\boldsymbol{\mu}_i) - X_{i,2})$ as output

residues are available for n_1 and n_2

$$r_{1,i} = \nu \| n_1(\boldsymbol{\xi}_i) \cdot n_2(\boldsymbol{\mu}_i) - X_i \|$$
$$r_{2,i} = (1-\nu) \| n_1(\boldsymbol{\xi}_i) \cdot n_2(\boldsymbol{\mu}_i) - X_i \|.$$

Only then, the concurrent learning will converge towards the real kinematics $k(\boldsymbol{\xi}, \boldsymbol{\mu})$.

6. Evaluation in simulation

This chapter documents the evaluation of learning sensorimotor maps with the previously presented novel model representation and associated learning algorithms as well as the new decomposition without intermediate markers. All experiments in this chapter are performed in simulation in order to confirm the theoretical properties of the applied methods. The chapter is structured into three parts. The first section evaluates the learning of kinematics models with the kinematic Bézier maps (KBM). In the following section, these algorithms are applied to learning the dynamics of a robot. Section 6.3 covers the improved learning with the decomposition of kinematic functions.

6.1. Evaluation of learning kinematic models

At first, the learning of kinematic functions with the KBM representation will be evaluated in combination with the associated learning algorithms. These are the standard linear least mean squares regression (LLMS), the incremental learning with the δ-rule, the nonlinear optimization of symmetry constraints and the partial least squares regression (PLS-1).

6.1.1. Setup

For the experiments, a generic simulated robot with six rotational degrees of freedom ($d_{\text{in}} = 6$) is defined by its parameters in the Denavit-Hartenberg formalism (see Figure 6.1):

$$a_i = 200 \text{ mm}, \quad d_i = 0 \text{ mm}, \text{ and } \alpha_i = 90^\circ \qquad \forall i \in \{1, \ldots, d_{\text{in}}\}. \tag{6.1}$$

The resulting robot has an overall length of 1.2 m when the robot is in its rest position, that is, all joint angles θ_i equal zero. Note that all possible joint configurations are valid in the simulations, that is, self-collisions are not considered.

For the training of the model and the evaluation of the interpolation, a restricted subspace of the configuration space $\Theta^{d_{\text{in}}}$ is defined in dependence of a wideness parameter δ

$$\Theta^{d_{\text{in}}}_{\text{intra}}(\delta) = \left\{ (\theta_1, \ldots, \theta_{d_{\text{in}}}) : \forall j \, |\theta_j| \leq \delta \right\} \tag{6.2}$$

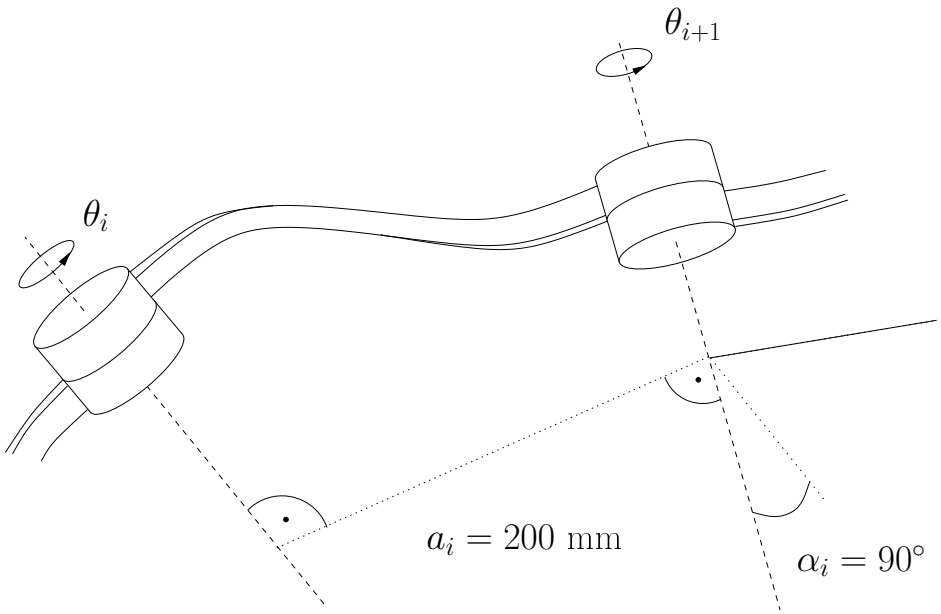
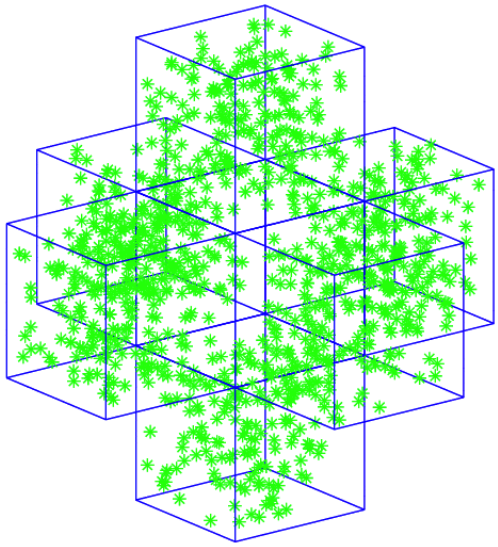
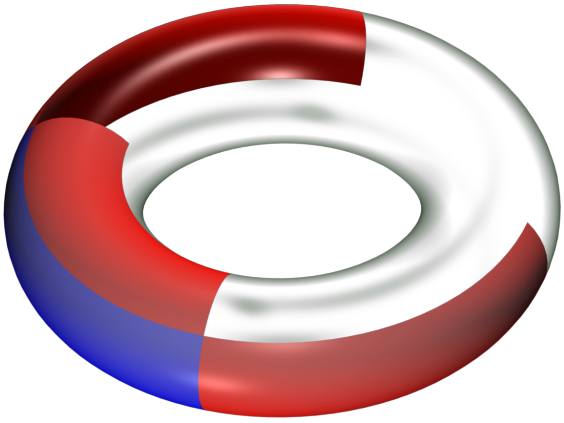


Figure 6.1.: Schematic of the simulated kinematics in this section. The image displays the rigid body transformation between adjacent joint axes by means of the Denavit-Hartenberg parameters. *Source: (Ulbrich et al., 2012c)*



(a) Uniformly distributed joint angles used for evaluating the exteroception for kinematics with three rotational degrees of freedom. The cube in the center is not sampled. From this region the data for the test sets is sampled.



(b) For a 2*R*-kinematics. The sets of possible training data points for training ($\Theta^2_{\text{intra}}(45°)$, *blue*) and evaluation ($\Theta^2_{\text{extra}}(45°)$, *red*) on the manifold describing the reachable workspace of a robot with two orthogonal joint axes. *Source: (Ulbrich et al., 2012c)*

Figure 6.2.: Selection of training and test data in the experiments.

such that the absolute values of the generated joint angles lie in an interval of size $2 \cdot \delta$. The associated training set with m samples is then defined as

$$\mathscr{T} = \left\{ (\boldsymbol{\theta}_i, \boldsymbol{x}_i) : \boldsymbol{\theta}_i \in \Theta_{\text{intra}}^{d_{\text{in}}},\ \boldsymbol{x}_i \in \mathbb{R}^3,\ i \in \{1, \ldots, m\} \right\} \tag{6.3}$$

where the $\boldsymbol{\theta}_i$ are randomly generated by a *uniform distribution*

$$\boldsymbol{\theta} \sim \mathscr{U}(\Theta_{\text{intra}}^{d_{\text{in}}}).$$

The end-effector positions $\boldsymbol{x}_i$ are generated by the simulated forward kinematics $f(\cdot)$. Artificial sensor noise in the joint encoders is simulated by adding a *normally distributed* error to all joint angles

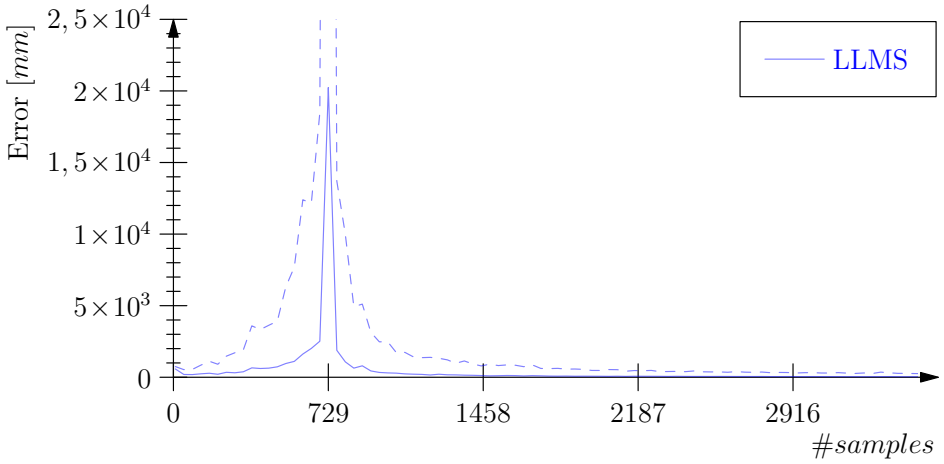
$$\boldsymbol{x}_i = f(\boldsymbol{\theta}_i + \boldsymbol{e}_i), \qquad e_{ji} \sim \mathscr{N}(0, \sigma)),\ j \in \{1, \ldots, d_{\text{in}}\} \tag{6.4}$$

Test data is generated analogously but *without* artificial sensor noise. That way, it can be evaluated if the learning can compensate the noise and produce more reliable predictions than the sensors can observe. This is possible because of the inductive bias when learning with KBM. The test data is therefore to be considered as *ground-truth* data which cannot be obtained without external high-precision sensors in an experiment with a real robot (see Chapter 7).

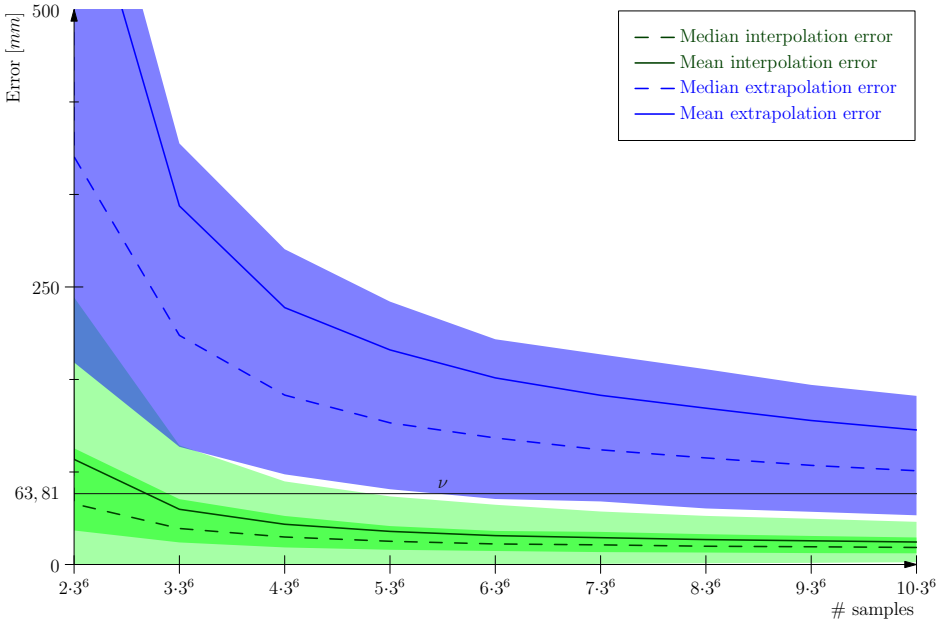
The experiments in this section also include the evaluation of the prediction of robot configurations in unexplored regions of the configuration space Θ^6 (that is, extrapolation). Therefore, the prediction errors in the neighboring regions of the configuration space used for training are considered. The neighboring regions are defined by the sub-space

$$\begin{aligned} \Theta_{\text{extra}}^{d_{\text{in}}}(\delta) &= \bigcup_{i}^{d_{\text{in}}} \left\{ (\theta_1, \ldots, \theta_{d_{in}}) : |\theta_j| \leq \delta\ \forall j \neq i \wedge |\theta_i| \leq 3 \cdot \delta \right\} \setminus \Theta_{\text{intra}}^{d_{\text{in}}}(\delta) \\ &= \bigcup_{i}^{d_{\text{in}}} \left\{ (\theta_1, \ldots, \theta_{d_{\text{in}}}) : |\theta_j| \leq \delta\ \forall j \neq i \wedge |\theta_i \pm 2 \cdot \delta| \leq \delta \right\}. \end{aligned} \tag{6.5}$$

This space is shown for three degrees of freedom in Figure 6.2a. The region is depicted as the composition of six areas indicated by their bounding boxes in the configuration space. This shows that the configuration space for the evaluation of the extrapolation is $(2 \cdot d_{\text{in}})$ times larger than the configuration space used for training. For two degrees of freedom, the reachable workspace of a robot with two rotational degrees of freedom is a torus. This allows an exemplary display of the respective configurations in the Cartesian space (see Figure 6.2b). In this image, the blue region indicates the set of possible training data while the red areas mark the set of possible samples used for evaluating the exteroception.
Again, the data generated for the evaluation of the exteroception has *no artificial noise* added to the joint angles.



(a) Learning with LLMS has a maximal prediction error for $3^{d_{\text{in}}}$ samples when overfitting is most severe. An artificial noise with standard deviation $\sigma = 3°$ is added to the joint angles.



(b) Close up of learning the errors when learning kinematics with fixed artificial noise ($\sigma = 2°$) and variable numbers of training samples. The bright and dark tinted regions indicate the standard deviation (std) and inter quartile ranges (iqr) respectively.

Figure 6.3.: Experiments with the standard LLMS learning with variable numbers of training samples and fixed noise.

6.1.2. Learning with linear least mean squares

The standard linear least mean squares regression (LLMS, see Section 4.1) is the most straight forward and most simplistic method for batch learning with KBM models. However, it is also the method that is most prone to overfitting (when there is sensor noise involved) unless enough training samples are given to compensate sensor noise, that is,

$$|\mathscr{T}| = m \gg 3^{d_{\text{in}}}.$$

Although it is possible to learn from less than $3^{d_{\text{in}}}$ samples resulting in the minimum norm solution, the prediction error of the learned model reaches its maximum for exactly $3^{d_{\text{in}}}$ samples, and reaches acceptable values for $m > 3 \cdot 3^{d_{\text{in}}}$ observations.

The first experiment demonstrates this by learning from samples generated with configurations out of a training set with a wideness parameter $\delta = 45°$ and a normally distributed noise added to each joint angle:

$$\mathscr{T} = \left\{ (\boldsymbol{\theta}_i, f(\boldsymbol{\theta}_i + \boldsymbol{e}_i)) : \boldsymbol{\theta}_i \in \Theta^{d_{\text{in}}}_{\text{intra}}(45°),\ e_{ji} \sim \mathscr{N}(0, 3°),\ \boldsymbol{x}_i \in \mathbb{R}^3,\ i \in \{1, \ldots, m\} \right\},$$

over a varying number m of training samples. The model is evaluated with data out of $\Theta^{d_{\text{in}}}_{\text{intra}}(45°)$ and $\Theta^{d_{\text{in}}}_{\text{extra}}(45°)$ for interpolation and extrapolation respectively without artificial noise and 500 samples each. The results are shown in Figure 6.3a and confirm the assumed characteristics of the learning curve. At $m = 729 = 3^6$, the maximal prediction error due to overfitting can clearly be seen. For higher number values of m the interpolation and extrapolation error drops quickly and continuously.

The focus of the second experiment lies on the observation of statistical characteristics of learning with redundant data (that is, $m \geq 3^6$). The setup resembles the first experiment with the exception that the artificial noise with standard deviation of $\sigma = 2°$ is less severe. Again, the model is evaluated for varying values of m and the test data for interpolation and extrapolation (3,000 samples each) is generated without artificial noise. The resulting curves are shown in Figure 6.3b. The artificial noise in the joint angles results in a position error v of 63.8 mm which is indicated in the image.
It can be observed that already with $3 \cdot 3^6$ samples, the mean prediction error for the interpolation drops below the position error v. Consequently, this means that the model now compensates the noise and predicts the position with a higher precision than the sensors are able to perceive.
The plot also shows that the mean and median of the prediction error do not coincide. This means that the error is not normally distributed. Furthermore only few outliers with high prediction errors occur.

The third series of experiments investigates the required quantities of samples necessary to compensate for different noise levels. Therefore, a variable standard deviation σ_i are inserted into Eq. (6.4) and the prediction error is evaluated for several KBM models learned from different numbers of training samples and two different pairs of restricted configuration spaces. The first configuration spaces used are

$$\Theta^6_{\text{intra}}(90°) \quad \text{and} \quad \Theta^6_{\text{extra}}(90°).$$

Although the value for δ is big, this means that only a small portion of $1/64$ of the complete configuration

space is used for the training and evaluation of the interpolation and a portion of $3/32$ for the extrapolation. The results of the mean prediction errors for extrapolation and interpolation in dependence of a variable noise

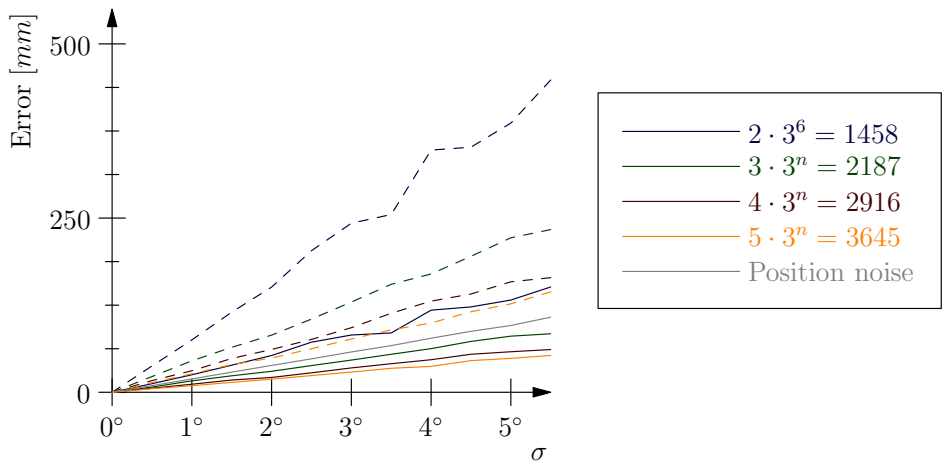
$$\sigma_i \in \{0°, 0.5°, 1°, \ldots, 5.5°\}$$

added to the joint angles are shown in Figure 6.4a. The first observation is that the position error due to the sensor noise in the joint angles is linear in the noise added to the joint angles. Second, the interpolation error again falls below the position error once a sufficient number $3 \cdot 3^6$ of training samples has been learned. It is important to note that this number remains constant for all tested degrees of angular noise. As a consequence, this means that any (reasonable) sensor noise can be compensated with this number of training samples under the evaluated conditions. However, the training samples used are randomly generated. Later experiments in Chapter 7 show that this does not hold if the training samples are too similar, for instance, if they are sampled along trajectories.
Figure 6.4b shows the same experiment repeated with different restricted configuration spaces

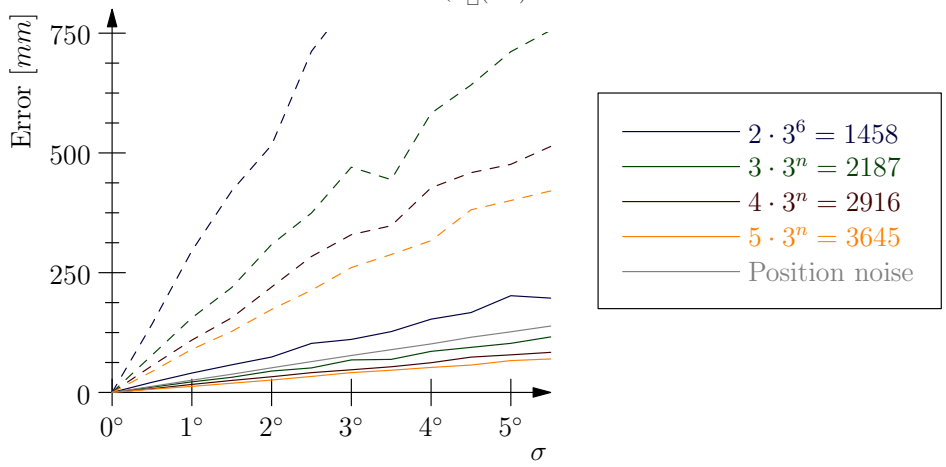
$$\Theta^6_{\text{intra}}(45°) \quad \text{and} \quad \Theta^6_{\text{extra}}(45°)$$

used for training and evaluation. The plot shows that the noise gains more influence such that less information can be extracted by learning. As a consequence, the quality of the extrapolation greatly degrades. The quality of the interpolation, however, remains mainly unaffected.

For the last experiment in this section, the previous experiment is repeated but this time, PSOM+ models (see Section 2.3.3) are learned instead of KBM models. PSOM+ models are an improvement of continuous Kohonen maps that take the global smoothness of the learned manifold into account which can be regulated by a hyper parameter λ. That way, learning with PSOM+ is a global learning method where each sample affects the outcome of the complete model. In this experiment, the models are defined over a neural lattice with 3^6 knots such that they have the same polynomial degree and computational complexity as the KBM used before. The models are learned with different smoothing factors and again with the same numbers of training samples and conditions as before. Results are shown in Figure 6.5. Unlike with KBM models, it is not possible to create an exact or even accurate model under these conditions. On the other hand, the impact of the noise is less significant on the prediction error because of the constant global curvature of the interpolation manifold. Compared to the classical PSOM, the extrapolation error is smaller by several magnitudes.
The prediction cannot be further improved for a lattice of this size, and the only way to resolve this problem is to resort to increasing the number of knots in the lattice. However, this leads to higher computational costs and increases the minimal number of training samples necessary to obtain a sufficient accuracy. In addition to this, the determination of the important smoothing factor λ is not trivial and has to be adjusted manually which complicates the application on a real system.



(a) The mean prediction errors for inter- and extrapolation are sufficiently low for well distributed samples ($\Theta^6_{\square}(90°)$.



(b) If the configuration space is further restricted ($\Theta^6_{\square}(45°)$) the quality of the extrapolation greatly declines.

Figure 6.4.: Experiments investigating the extrapolation capabilities of the standard LLMS. Continuous lines indicate the mean errors of the interpolation and dashed line those of the extrapolation.

6.1.3. Nonlinear optimization of symmetry constraints

Now, the nonlinear optimization that also considers the symmetry of the control net of the KBM models is evaluated (see Section 4.4). Therefore, three experiments very similar to the experiments of the previous section are carried out and only the learning algorithm is replaced. The optimization is an iterative method. The maximal number of iterations is an additional hyper parameter and is set to a relative small limit of 150 iterations to speed up the evaluation.

First, training samples and test data are generated over

$$\Theta^6_{\text{intra}}(90°) \quad \text{and} \quad \Theta^6_{\text{extra}}(90°).$$

and the model is initialized with the solution of the standard LLMS before the optimization (see Figure 6.6a). As the initialization already has a good quality, the optimization is able to greatly improve the prediction. It is now even possible that the extrapolation error falls below the position noise. In opposition to the standard

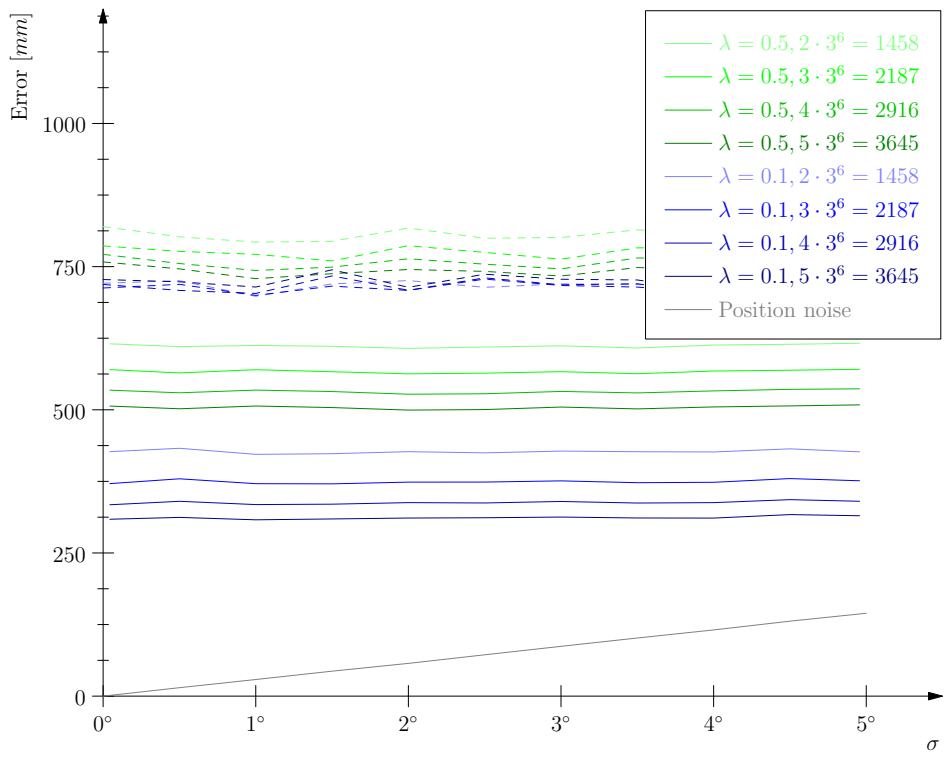


Figure 6.5.: Experiment evaluating the extrapolation of the PSOM+ under conditions similar to those used in Figure 6.4. Continuous lines indicate the mean errors of the interpolation and dashed line those of the extrapolation.*Source: (Ulbrich et al., 2012c)*

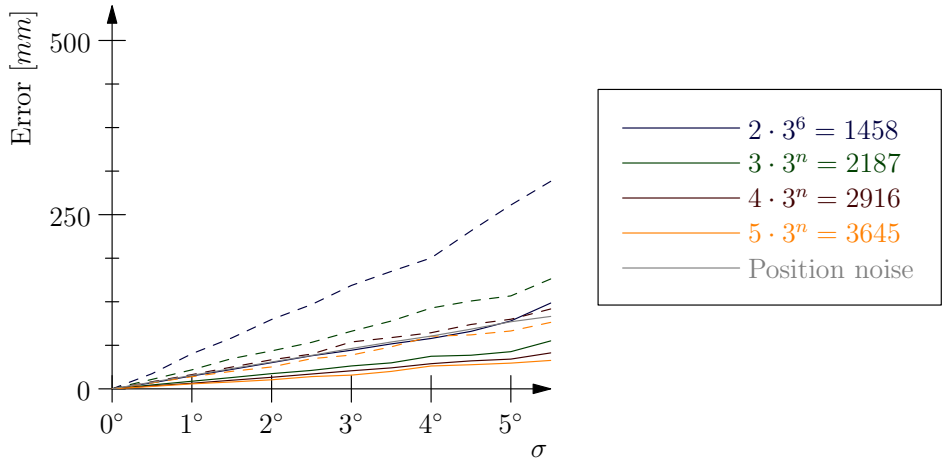
LLMS, the nonlinear optimization has a hyper parameter ν that balances the symmetry optimization and the adaptation to the training data. In this experiment, ν has was set to a value of 1.0 (that is, that both optimizations are equally weighted) and has no critical influence on the result.

The second experiment uses data generated from the smaller configuration spaces

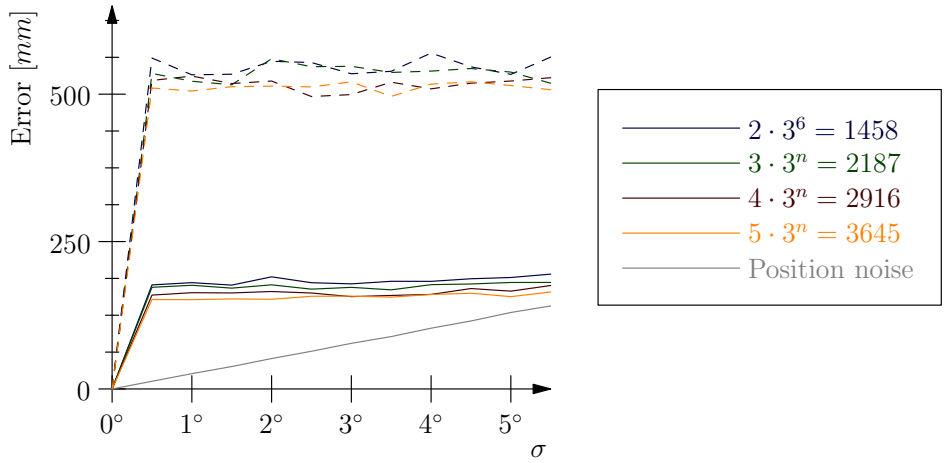
$$\Theta^6_{\text{intra}}(45°) \quad \text{and} \quad \Theta^6_{\text{extra}}(45°).$$

Again, the model is initialized with the standard LLMS solution. This time, however, the model is degenerated because of the higher influence of the sensor noise and the algorithm does not converge to a valid solution within the maximal number of iterations as shown in Figure 6.6b. Further, the weighting factor ν had to be reduced to 0.5 (giving more influence to the optimization of the prediction error) to achieve the best results. This behavior can be explained by the deformation of the control net that results from the least mean squares solution where the circles in the main directions of the manifold become ellipses. If the net is too strongly deformed, the optimization gets caught in a local minima. The necessity to decrease the influence of the symmetry constraints results from the joint configuration that lie closer together and therefore provide less information about the shape of the manifold. As a consequence, the influence on the adaptation to the training data has to be increased to achieve the best results under these conditions.

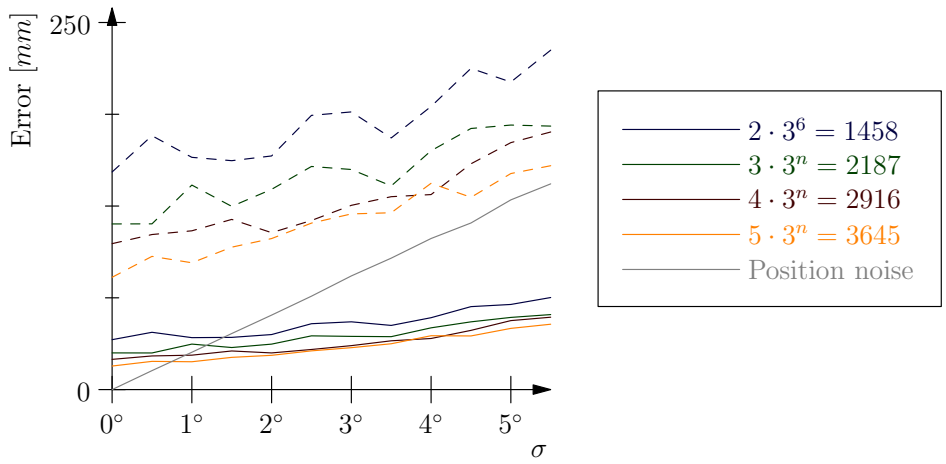
In the last experiment, the model uses the same data but is initialized by a linear approximation to the training data (see Section 4.3.2) and the learn rate has been further decreased to 0.1. The results are shown



(a) Optimization of a model learned with the standard LLMS when training data is well distributed $\Theta^6_\square(90°)$ and the weighting factor is $\nu = 1.0$.



(b) Same as above when the data lies closer together ($\Theta^6_\square(45°)$). The weighting factor has to be reduced to $\nu = 0.5$ to achieve the better results.



(c) Starting with a linear initialization leads to much better results for $\Theta^6_\square(45°)$. The weighting factor is $\nu = 0.1$.

Figure 6.6.: Experiments with the nonlinear optimization of symmetry constraints. Continuous lines indicate the mean errors of the interpolation and dashed line those of the extrapolation.

in Figure 6.6c. This experiment gives the best results in this setup. For zero noise, the prediction error is not zero as in the previous experiments with the KBM as the linear initialization differs greatly from the perfect model created with zero noise and learning has to converge to the LLMS solution. Remember that the optimization is configured to stop after 150 steps and the final convergence is slow close to the target. On the other hand, the learning proved to be very tolerant to sensor noise and the prediction error grows so slowly that it quickly falls significantly below the position noise (even for the extrapolation). This is a property of the maintained symmetry and is a characteristics shared with the PSOM+.

Altogether, these are remarkable results. However, it has to be noted that finding the optimal combination of hyper parameters and initialization is not trivial and requires manual adjustment and experience. Further, the nonlinear optimization requires by far the most complex implementation of the algorithms presented in this thesis and has also the slowest execution. These aspects limit its application on real system.

6.1.4. Learning with partial least squares regression

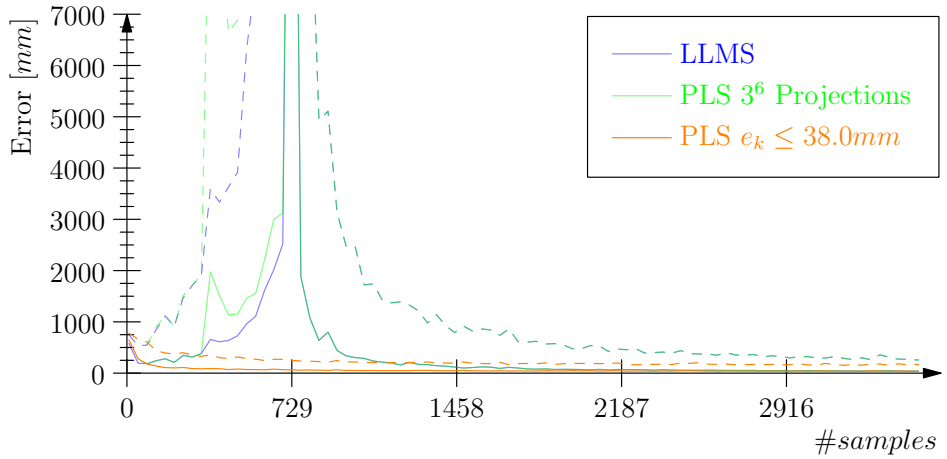
The one-dimensional partial least squares regression (PLS-1, see Section 4.2) is an improvement of the standard LLMS. It can either be configured to find the absolute minimum least squares solution or to stop once the prediction error over the training data falls below a threshold $e_{\max}$. Although $e_{\max}$ is an additional hyper parameter that has to be manually adjusted according to the training data, the value directly corresponds to an observable statistics of the training data. In simulation or with external sensors, this value can be directly measured from the position error due to the sensor noise.

The first experiment in this section is a reproduction the first experiments of the last section. The performance of the standard LLMS is compared to the PLS-1. Learning with PLS-1 is repeated twice—first with a value of zero for $e_{\max}$ (that is, the algorithm terminates after the complete 3^6 steps) and then using the threshold $e_{\max} = 38mm$. This value is the observed prediction error for the individual coordinates of the end-effector and causes the algorithm to terminate after fewer steps preventing overfitting. The results are displayed in Figure 6.7a. As expected, the curves of the standard LLMS and the PLS-1 with $e_{\max} = 0$ unite at 729 training samples. Both algorithms then compute the unique least mean squares solution and are subject to the same degree of overfitting. However, this does not happen to the second PLS-1 with a non-zero threshold. The algorithm efficiently models the noise and the prediction error decreases the more training data is processed. This reduces the lower bound of training data required to reach a good accuracy.
This phenomenon can be explained by the redundancy that is contained in a well-formed control net which is implicitly detected by the algorithm.

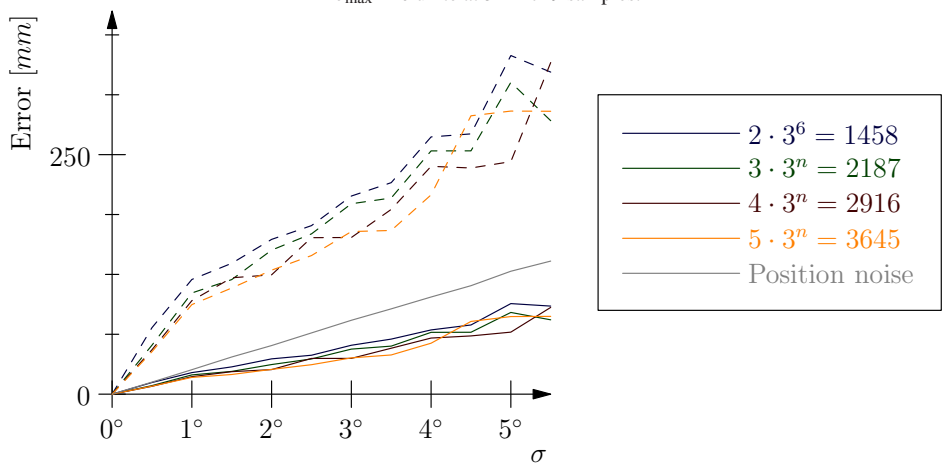
The second experiment uses the restricted configuration spaces

$$\Theta^6_{\text{intra}}(45^\circ) \quad \text{and} \quad \Theta^6_{\text{extra}}(45^\circ)$$

to evaluate the impact of different degrees of angular noise on the prediction in analogy to previous experiments. In addition to the calculation of the position noise, the mean prediction error for each coordinate of



(a) Comparison of learning kinematics with standard LLMS and PLS-1 with $e_{\max} = 0$ and $e_{\max} = 38mm$. A constant artificial noise of $\sigma = 3°$ is added to the joint angles. The curves of the LLMS and PLS-1 with $e_{\max} = 0$ unite at $3^6 = 729$ samples.



(b) Analysis of extrapolation and interpolation of models learned with PLS-1 and a dynamic $e_{\max}$ extracted from the data. Continuous lines indicate the mean errors of the interpolation and dashed line those of the extrapolation.

Figure 6.7.: Experiments with partial least squares. The data is distributed in $\Theta^6_{\square}(45°)$.

the end-effector is computed for each degree of noise. This value is then set used as $e_{\max}$ in the respective computation. The plot of the results presented in Figure 6.7b unveils remarkably good results. It can be seen that the difference between the prediction errors of models learned with different numbers of training samples is very little. This shows that the PLS requires significantly less samples compared to the standard LLMS and proves its robustness to sensor noise.

Although, the symmetry optimization yields slightly better results for $\Theta^6_{\text{intra}}(45°)$, learning with PLS-1 proved to be the most stable and efficient regression algorithm for learning the KBM representation. It only depends on a single hyper parameter with an meaningful relation to the training data. Reversely, when fine tuning of this parameter becomes necessary the determined value can be used to assess the sensors of the system.

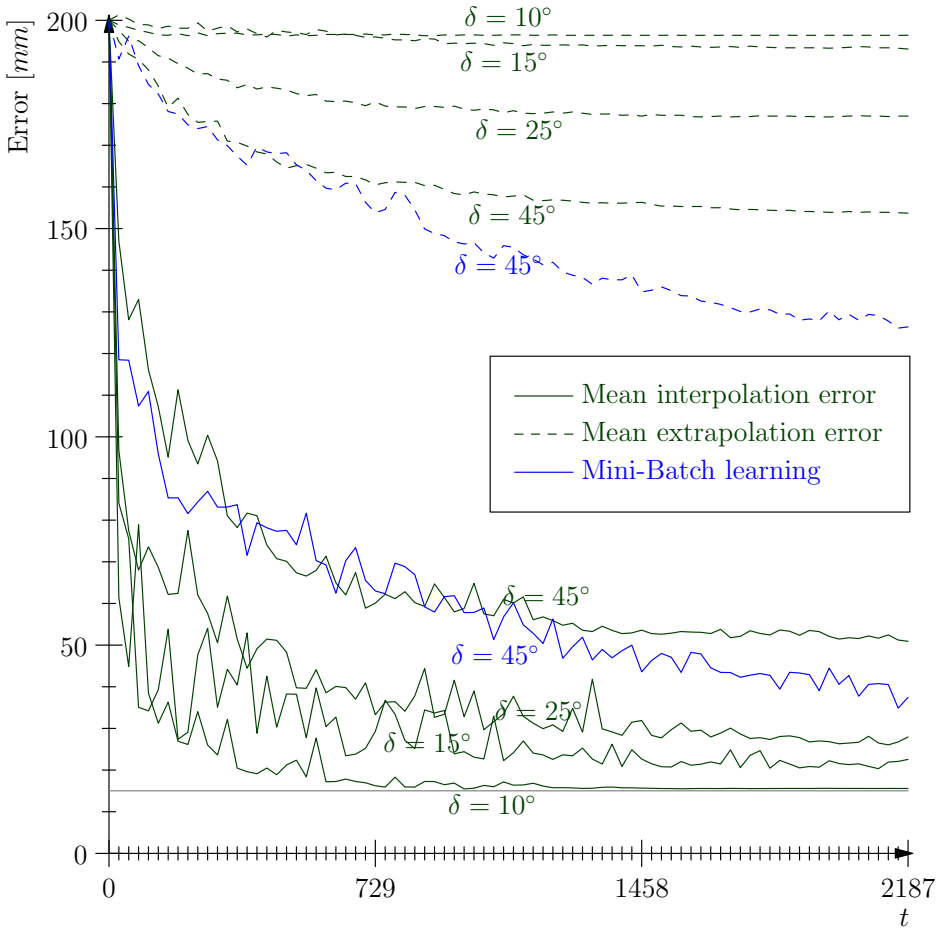


Figure 6.8.: Experiments with incremental learning of the KBM model. the adaptation to a tool of 200 mm length is simulated. Multiple models are learned and evaluated over differently restricted configuration spaces.

6.1.5. Incremental learning with the δ-Rule

The δ-rule (see Section 4.3) is a minimalistic approach to incrementally learn a KBM model. Therefore, it is very well-suited to quickly adapt to rapid changes of the kinematics although the changes mostly affect the model locally. This experiment simulates the case of the robot holding a tool. The last arm element of the robot is therefore elongated by 200 mm simulating a stick-like object held by the end-effector to increase its range. The Denavit-Hartenberg parameters of Eq. (6.1) are adjusted accordingly

$$a_6 = 400 \text{ mm}, \quad d_6 = 0 \text{ mm}, \text{ and } \alpha_6 = 90^\circ. \tag{6.6}$$

Normally distributed artificial noise with a standard deviation σ of 2° is added to the joint angles as in previous experiments. Learning with the incremental method is significantly slower compared to batch learning. For this reason, the model is initialized with an 'ideal' model of the original kinematics learned with the standard LLMS regression from data with zero noise. In these experiments the wideness parameter δ for the partial configuration spaces

$$\Theta^6_{\text{intra}}(\delta) \quad \text{and} \quad \Theta^6_{\text{extra}}(\delta)$$

is varied to confirm the hypothesis that incremental learning quickly improves the local prediction.
Figure 6.8 shows the results of this experiment. For small values of δ, the prediction error drops in fact very fast and quickly reaches the position noise of 15 mm. Bigger values of δ lead to a much slower convergence and stagnate at a lower precision. On the other hand, the bigger the value of δ the stronger is the positive effect of the incremental learning on the extrapolation error. These observations confirm the hypothesis of the local influence.
The curve labeled with 'mini-batch learning' represents a variant of the algorithm where a small number of training samples (27 in this experiment) is collected and repeatedly processed by the incremental learning. By this repetition, the data samples gain a stronger influence on the model. This compromise between batch and incremental learning results in faster convergence. The higher the number of samples used for the mini batches is, the stronger is this effect.

6.2. Evaluation of learning dynamic models

This section presents the evaluation of the extension of the KBM to learn the inverse dynamics of a robot with serial joints.

6.2.1. Setup

The experiments in this section are carried out with the *Robotics Toolbox for MATLAB* (Corke, 2011) which features a dynamic simulation for rigid body dynamics. Therefore, a dynamic model of the industrial *PUMA-560 (Programmable Universal Machine for Assembly)* robot manipulator is applied. Recall that learning the inverse dynamics with the KBM variant requires significantly more training data compared to learning the kinematics. For this reason, the simulated robot has a reduced set of four controllable degrees of freedom (that is, the most distal joints are ignored), in order to be able to work with a reasonable amount of training samples. In addition, the effects of friction are not simulated and neglected as they cannot be directly included in the model. On a real platform, the friction has to be individually modeled for each joint and included directly into the control module.

6.2.2. Batch learning

In the first experiments, it is investigated if the inverse dynamics represented by the modified KBM representation presented in Section 3.2.2 and learned with the standard LLMS regression expose the same properties with respect to tolerance to sensor noise as in the case of learning the kinematics. The experiments are therefore conducted under similar conditions as described in the previous section.

At first, the number of training samples required to absorb a constant noise is examined. Compared to learning the kinematics, more data is required for learning subsumed in the generalized coordinates. The

generalized input values to learn the inverse dynamics are the current joint configuration $\boldsymbol{\theta}_i$, joint velocities $\dot{\boldsymbol{\theta}}_i$ and joint accelerations $\ddot{\boldsymbol{\theta}}_i$. The generalized outputs are the joint torques $\boldsymbol{q}_i$. Unlike in previous experiments, the artificial noise is now added to the output values in form of a normally distributed noise with zero mean a standard deviation of 5% of the expected value ranges of the individual joint torques. Altogether, the training set for this experiment is defined as

$$\mathscr{T}(m) := \Big\{ (\boldsymbol{\theta}_i, \dot{\boldsymbol{\theta}}_i, \ddot{\boldsymbol{\theta}}_i, \boldsymbol{q}_i),\ \boldsymbol{\theta}_i \in \Theta^4_{\text{intra}}(45^\circ), \\ \dot{\boldsymbol{\theta}}_i \in [-0.25, 0.25]^4,\ \ddot{\boldsymbol{\theta}}_i \in [-1,1]^4,\ \boldsymbol{q}_i \in \mathbb{R}^4,\ i \in \{1,\dots,m\} \Big\},$$

over a variable number m of traing samples. The artificial noise is introduced in

$$\boldsymbol{q}_i = f(\boldsymbol{\theta}_i, \dot{\boldsymbol{\theta}}_i, \ddot{\boldsymbol{\theta}}_i) + \mathscr{N}(0, \boldsymbol{\sigma})$$

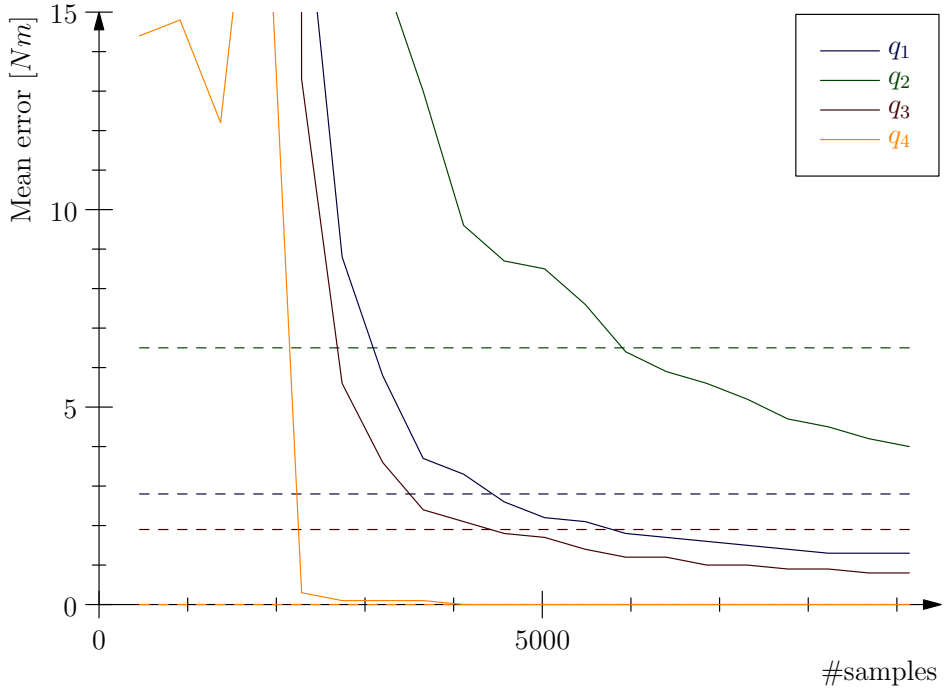
given the simulation of the inverse dynamics as a function $f(\cdot)$.
Multiple models are trained with an increasing size m of the training set. After learning, the data is evaluated against equally created test data (2,000 samples) with the exception that no artificial sensor noise is added. Again, this experiment shows whether the model can absorb the noise and how many training observations are required therefore.
The results can be examined in Figure 6.9a. They confirm that the noise is successfully absorbed when m exceeds three to four times the number of the model parameters (1,831, see Table. 3.1). For more redundant data, the prediction error falls below the sensor error induced by the artificial noise.

The second experiment demonstrates how models learned with fixed-sized training sets absorb a variable degree of noise. The artificial noise added to the joint torques is constantly increased from 0% to 5% of the expected value ranges of the respective joint torques. In addition, the extrapolation error over joint configurations not observed during training is examined. The restricted configuration space used to evaluate the extrapolation is $\Theta^4_{\text{extra}}(90^\circ)$. The angles of the test data are equally distributed. For the velocities and accelerations extrapolation sets are defined analogously to Eq. (6.5):

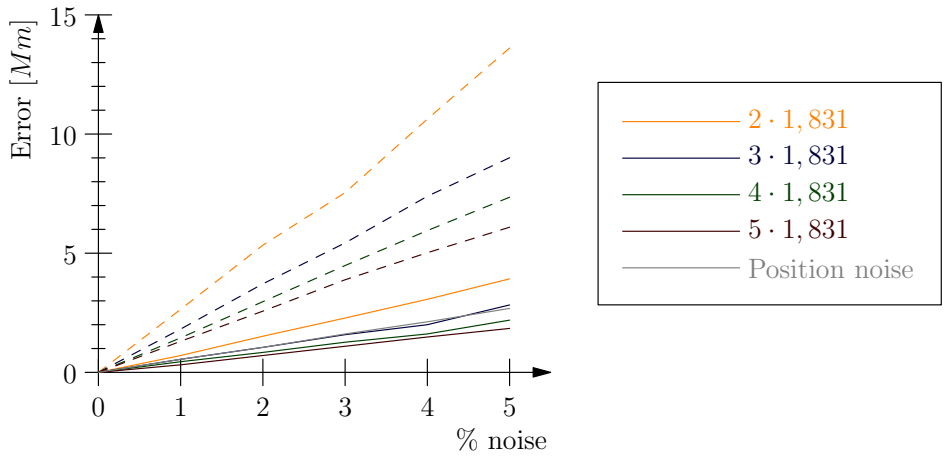
$$\dot{\Theta}^4_{\text{extra}}(0.25) = \bigcup_i^4 \Big\{ (\dot{\theta}_1, \dots, \dot{\theta}_4) : |\dot{\theta}_j| \leq 0.25\ \forall j \neq i \wedge |\dot{\theta}_i \pm 2 \cdot 0.25| \leq 0.25 \Big\}.$$

$$\ddot{\Theta}^4_{\text{extra}}(1.0) = \bigcup_i^4 \Big\{ (\ddot{\theta}_1, \dots, \ddot{\theta}_4) : |\ddot{\theta}_j| \leq 1.0\ \forall j \neq i \wedge |\ddot{\theta}_i \pm 2 \cdot 1.0| \leq 1.0 \Big\}.$$

The results displayed in Figure 6.9b illustrate that the KBM can absorb any reasonable degree of sensor noise given enough training data similar to the case of learning the kinematics. The prediction error caused by the noise itself is again depicted and it can be seen that for more than three times of the number of parameters the prediction of the model is more accurate than the sensors. Further, the extrapolation of the models allows then predictions for unexplored configurations with an acceptable error.

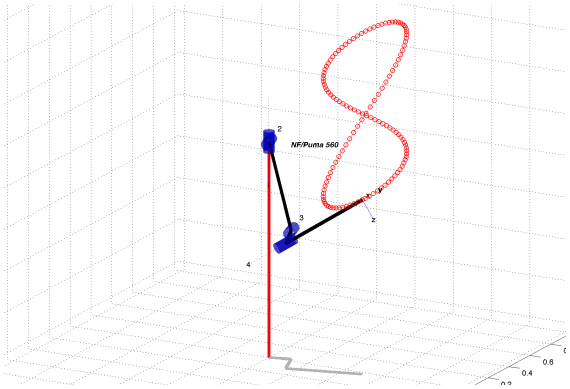


(a) A constant artificial sensor noise added to the joint torques with a standard deviation of $\sigma = 5\%$ of the expected value range can be absorbed after three times the number model parameters.

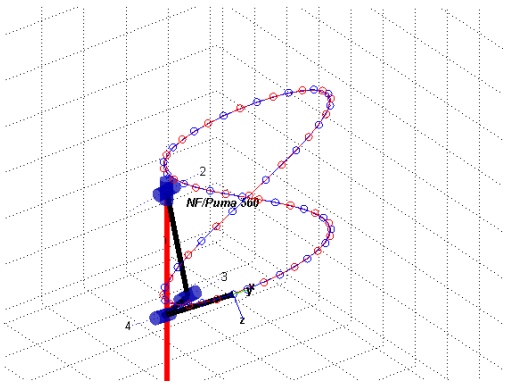


(b) For different degrees of artificial sensor noise in the joint torques, the prediction error for interpolation and extrapolation are shown for models learned with fixed numbers of training samples. Continuous lines indicate the mean errors of the interpolation and dashed line those of the extrapolation.

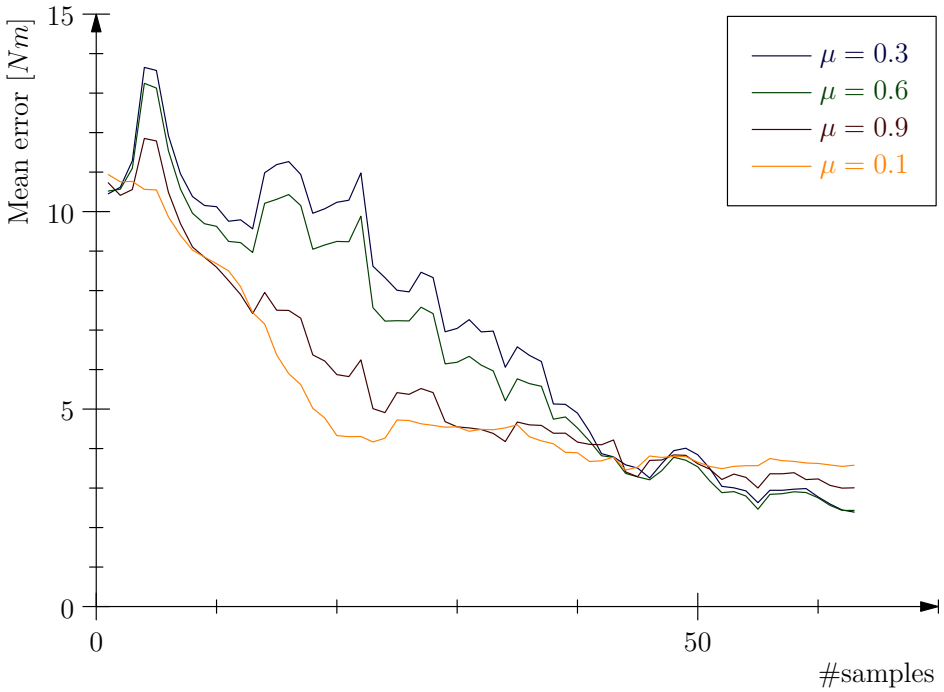
Figure 6.9.: Experiments for batch learning of the inverse dynamics with the standard LLMS.



(a) The ‘figure eight‘ trajectory the robot executes during the experiment.



(b) The recorded data is alternatingly divided into test *(blue)* and training samples *(red)*.



(c) The robot adapts to holding a heavy tool. Four KBM models were initialized with an ‘ideal’ of the inverse dynamics model pior to tool use. Then they were trained to absorb the tool. Four different learn rates are examined.

Figure 6.10.: Experiment for incrementally learning the inverse dynamics.

6.2.3. Incremental learning with the δ-Rule

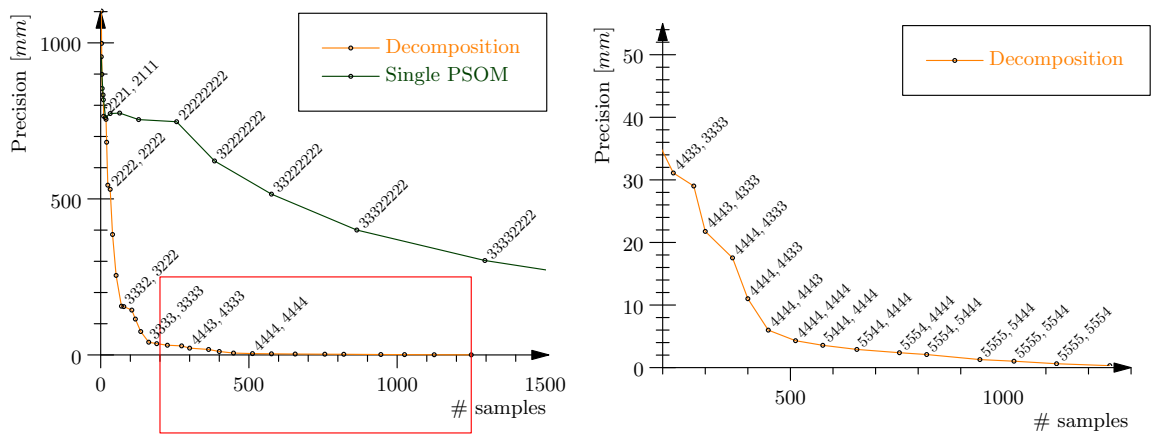
This paragraph describes the evaluation of incrementally learning the inverse dynamics with the KBM models and the δ-rule (see Section 4.3). One of the most useful applications of incremental learning is when an inaccurate or outdated model is available a priori and can be refined by the algorithm. The experiment simulates that the robot holds a heavy tool, like a hammer for instance. No artificial noise is used in this experiment and a perfect model of the dynamics prior to holding the tool is created by batch learning. Then a heavy object with a length of 20 centimeters and a weight of 25 kilograms is attached to the end-effector. The center of mass is located in the tip of the simulated tool. This object influences greatly the dynamics of the robot.
This time, the data for this experiment is not generated from uniformly distributed samples. Instead, the observations are made while the end-effector constantly moves along a periodic trajectory. The trajectory is called *'figure eight'* and is described by the function

$$g(\cdot): \quad [-2\cdot\pi, 2\cdot\pi] \to \mathbb{R}^3, \quad \alpha \mapsto (0.4\cdot\sin\alpha, -0.5, 0.4\cdot\cos\alpha/2 + 0.2)^T,$$

where the parameter α varies between $(-2\cdot\pi)$ and $(2\cdot\pi)$ within ten seconds. This motion generates the joint positions, velocities and accelerations as well as the torques required for refining the model. The trajectory is shown in Figure 6.10a together with the visualization of the simulated robot. When the robot executes the motion, 42 value pairs of generalized coordinates are recorded during a single period. The values are alternatingly divided into training and test samples as shown in Figure 6.10b. The experiments terminates after three repetitions of the periodic movement. Multiple models were refined. Each was equally initialized but learning was performed with different learn rates. That way, the best value for this hyper parameter can be determined.
The results in Figure 6.10c reveal that the prediction error reduces surprisingly quickly in this scenario—already during the first repetition. Further, it shows that lower adaptation rates initially lead to better adaptation while these decrease in the long term. These observations can be explained by the distribution of the training samples. In the previous experiments with incremental learning, the training and test data was randomly distributed over a wide range of the configuration space. Here the samples are very similar to each other which is beneficial for the incremental learning that has a strong local effect. The similarity of the training samples also explains why a lower adaptation rate initially leads to better results while higher rates show a mild effect of overfitting.
This experiment demonstrates how rapidly the KBM can be incrementally adapted in a realistic scenario. That way, it is possible to quickly adapt to holding a tool for a well-defined action without the need of exploring a big portion of the configuration space as it were necessary for batch learning.

6.3. General decomposition without intermediate markers

In this section, the novel general decomposition without intermediate markers is evaluated. The experiments use simulated robot kinematics as defined in Eq. (6.1) with either eight or twelve controllable rotational



(a) Comparison of learning the decomposition and learning a single network. The labels of the data points indicate the number of neurons per dimension in the neural layer.

(b) Closeup of the results of the decomposition.

Figure 6.11.: Experiments for learning the decomposition with PSOM and batch learning.

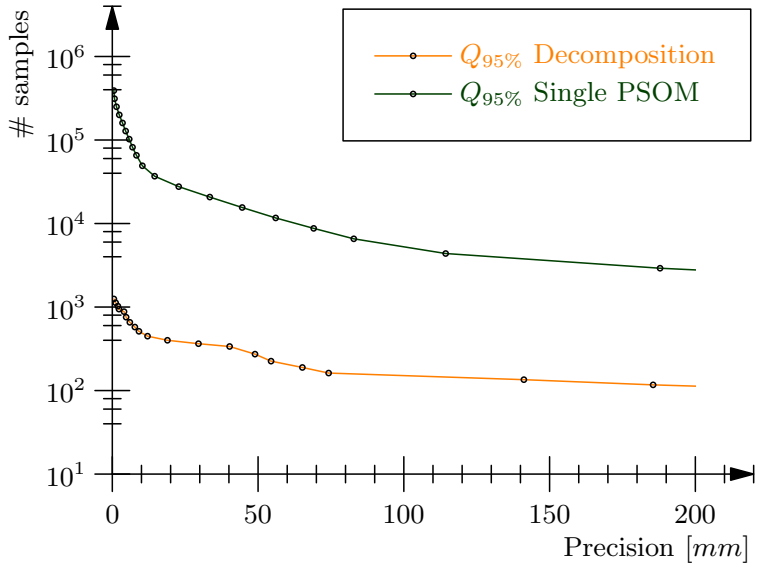


Figure 6.12.: The same experiment as shown in Figure 6.11. The axes are swapped and the number of samples is shown on a logarithmic scale.

degrees of freedom. Training samples used for training and evaluation are generated by joint configurations out of

$$\Theta^{d_{\text{in}}}_{\text{intra}}(45°),$$

and only the prediction error for the interpolation is determined in the experiments and no artificial noise is applied in the experiments.

6.3.1. Offline learning

For the experiments evaluating the decomposition with offline learning, two general function regression methods—namely the *parameterized self-organizing maps (PSOM)* and *Gaussian mixture regression (GMR)*—are applied. This is because the decomposition decomposes the forward kinematics into partial functions that are valid kinematic functions for themselves. It is obvious that the partial kinematics can be represented and learned with zero prediction error by KBM models as noise is neglected in these experiments. For this reason, the KBM are only applied in the context of online learning of the partial kinematics.

In the first experiment, a simulated robot with eight independent degrees of freedom is used. The partial kinematics are learned by PSOM models. The weights of the neurons are arranged in an eight-dimensional neural lattice. The coordinates in the lattice directly correspond to joint angles, and the weights of the neurons are directly assigned to the respective output of the forward kinematics (that is, the workspace of the robot is regularly sampled, see Section 2.3.3). During the experiment, the networks are learned with different numbers of neurons n and the prediction error is observed:

$$n = n_1 \cdot \ldots \cdot n_8,$$

where the n_i are the numbers of neurons per dimension in the neural layer.
That way, it becomes possible to compare the numbers of training samples (that directly corresponds with the number of neurons) required to reach the same accuracy for networks learned with and without the decomposition. The results are shown in Figure 6.11a and Figure 6.11b. As expected, the mean prediction error of the estimate using the decomposition decreases much quicker compared to the estimate of the single network. Now the magnitude of the reduction is investigated. Therefore, Figure 6.12 shows a different view on the results. In this plot, it can be seen how many samples are required for learning to reach a given precision. The number of samples is shown on a logarithmic scale.
Given that a single network requires

$$n_s = q^{d_{\text{in}}}, q \in \mathbb{R}$$

samples to reach a given precision, a decomposition into m partial kinematics should require

$$n_d = m \cdot q^{d_{\text{in}}/m}$$

samples to reach the same precision in theory. The experiment confirms this hypothesis. Given the numbers n_s and n_d to learn the single network and the decomposition respectively, the ratio between their logarithms has to be

$$\frac{\log n_s}{\log n_d} \approx \frac{\log n_s}{\log \sqrt{n_s}} = 2.$$

In fact, this relation can be observed in Figure 6.12. The curve for the single network approximately equals the curve of the decomposition scaled along the ordinate by the factor of two. The high accuracy of this relation is shown in Table. 6.1 directly on the collected data providing the strong evidence for the correctness of the hypothesis.

Precision [mm]	n_s	n_d	$\frac{ln(n_s)}{ln(n_d/2)}$
1,100	1	2	
760	16	10	1.72
520	576	32	2.29
390	864	40	2.26
270	1,296	52	2.2
140	2,916	105	2.01
40	8,748	162	2.07
30	11,664	225	1.98
20	20,736	300	1.98
10	36,864	400	1.98
4.2	65,536	512	2.00
1.3	200,000	945	1.98
0.3	390,625	1,250	2.00

Table 6.1.: Tabular view on the data shown in Figure 6.11. The ratio of the logarithms of the respective numbers of samples is shown in the last column.

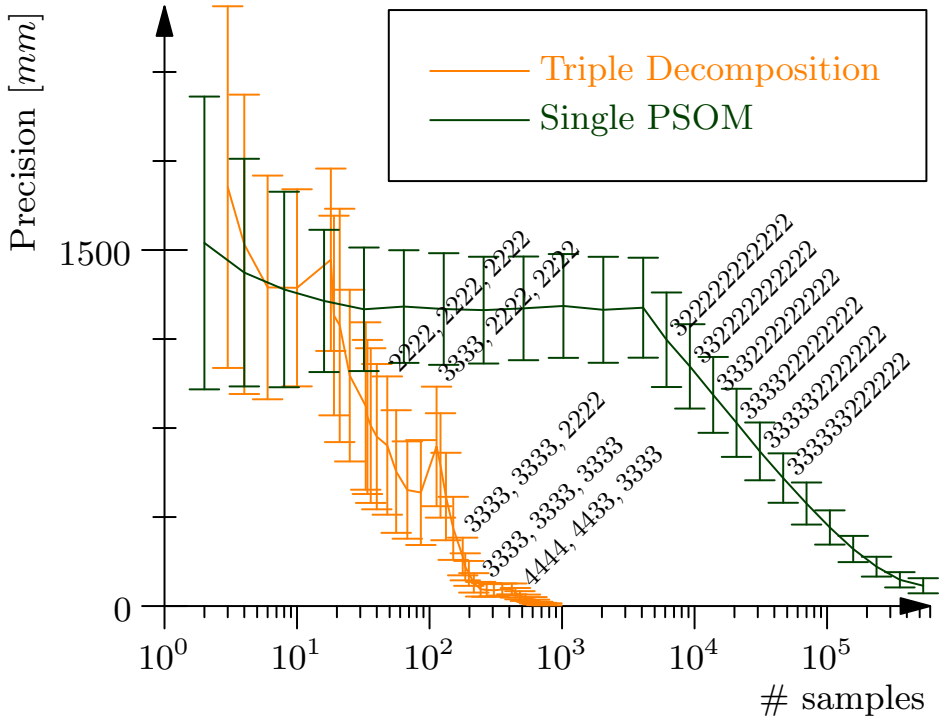


Figure 6.13.: Experiment with a hyper redundant robot with twelve degrees of freedom. The error bars indicate the standard deviation of the prediction error.

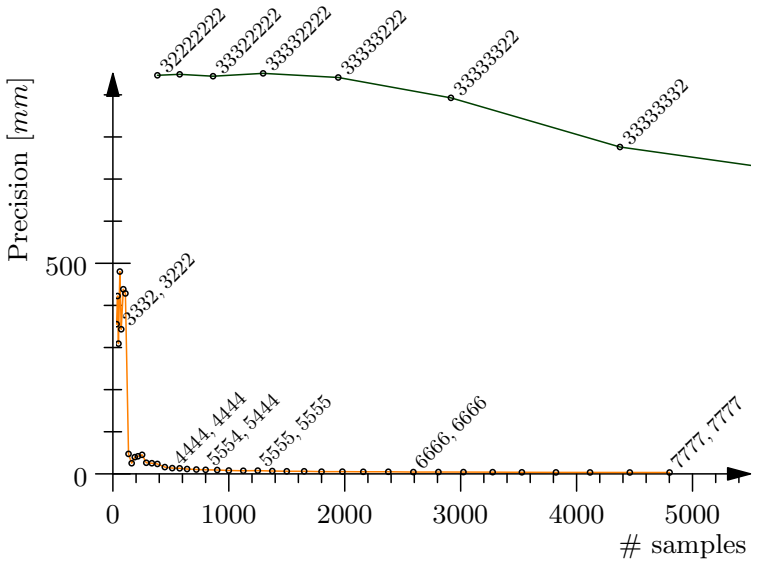


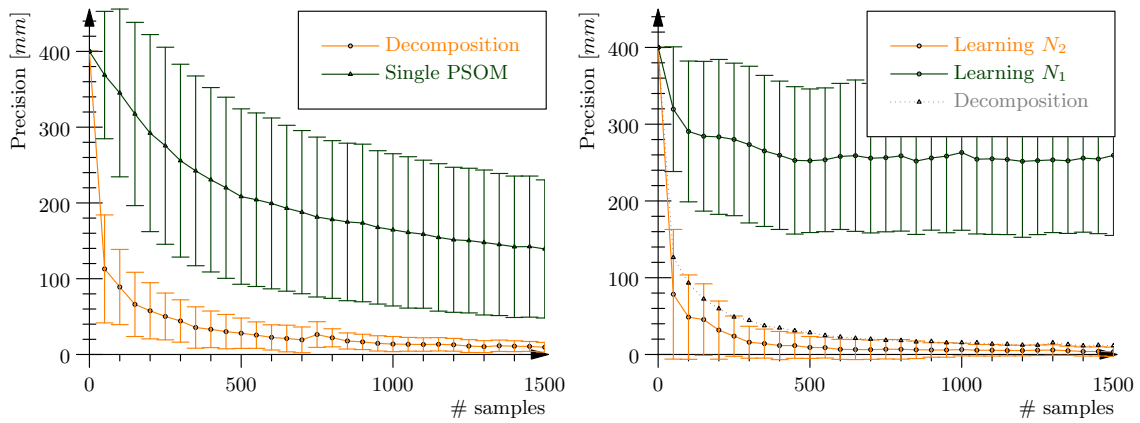
Figure 6.14.: An Experiment comparable to Figure 6.11 but using an GMM as an alternative learner. As the same number of samples are generated as for the PSOM, the same labels of the data points are used.

The next experiments aims at demonstrating the scalability of the decomposition approach. Therefore, a hyper redundant robot with twelve independent degrees of freedom is learned with a recursive decomposition. The decomposition is applied such that three partial kinematics of equal length can be learned efficiently and renders learning affordable. The results are shown in Figure 6.13. The same precision obtained by a single network after learning $1 \cdot 10^6$ samples is reached with the triple decomposition already after 110 examples proving the efficiency of the approach.

In the last experiment, an alternative learner—the Gaussian mixture regression (GMR)—is applied instead of the PSOM. Learning with PSOM has been very fast because of the direct interpolation of the observed output of the forward kinematics. GMR requires significantly higher numbers of training samples to train the local models. On the other hand, it can be trained with randomly distributed samples whereas the PSOM required the workspace to be regularly sampled. In addition, the GMR is more robust to sensor noise and stores its acquired knowledge in a very compact form (instead of storing all samples). To make the results of this experiment comparable to previous results, the length of each arm element has to be reduced to its half (100 mm) and the configuration space is further restricted and training data is generated randomly with configurations out of

$$\Theta^6_{\text{intra}}(22.5^\circ).$$

The only hyper parameter required to be adjusted for the GMR is the number of mixture models. It turned out that this parameter is approximately proportional to the number of samples. It is set to its optimum in the experiment determined manually in multiple trials. The results shown in Figure 6.14 suggest that the speedup provided by the decomposition is similarly obtained with the GMR.



(a) Learning both networks simultaneously.

(b) Learning only one of the networks. The result of (a) is shown in gray for comparison.

Figure 6.15.: Experiments with simulated tool use. The error bars indicate the standard deviation of the prediction error.

6.3.2. Online learning

Now, it is evaluated how a previously learned decomposition can adapt to structural changes of the kinematics by incremental learning. Therefore, two scenarios of common deformations are examined. The first represents tool use. To simulate that the robot holds a tool, the length of the last arm element is elongated to its double of 400 mm. The second scenario occurs, for instance, if a robot is equipped with incremental encoders only. Such a robot has to be initialized by moving to its joint limits before application. As a consequence, the encoder values are often inaccurate which can be modeled by a fixed offset $\boldsymbol{\alpha}$ to the joint angles

$$\bar{\boldsymbol{\theta}}_i = \boldsymbol{\theta}_i + \boldsymbol{\alpha}.$$

Such a 'shift' in the joint angles is also likely to happen in tendon-driven robots after repair. For the experiments, a constant offset of

$$\boldsymbol{\alpha} = (10^\circ, \ldots, 10^\circ)^T$$

has been chosen.

At first the two scenarios are evaluated again with the PSOM representation. To implement an incremental learning scheme, the δ-rule (see Section 4.3) is applied. This adaptation rule can be applied to any global representation with an underlying polynomial basis such as the KBM and the PSOM and it converges to the least mean squares solution. Such incremental learning reacts instantly to deformations but converges slowly and requires more training samples as compared to batch learning. For the PSOM, the length of the kinematic chain is therefore reduced to five arm elements and independent degrees of freedom.

The results of the experiment covering simulated tool use are shown in Figure 6.15. For the experiment in Figure 6.15a, the adaptation rule was applied to both network simultaneously with an adaptation rate of 0.5 (see Chapter 5). The prediction error of the decomposition drops much quicker but also does the standard deviation (marked by the error bars) which nearly does not decrease at all for the single network. Tool use

is a deformation that only affects the last arm element. For this reason, it has been investigated how the adaptation to the deformation behaves when applying the incremental learning to one network only. On the one hand, when adapting the network that depends on the proximal joints, it is expected that the deformation cannot be compensated at all. On the other hand, the adaptation should be much faster when adapting the remaining network only. These hypothesis could be confirmed in an experiment. The results can be seen in Figure 6.15b.

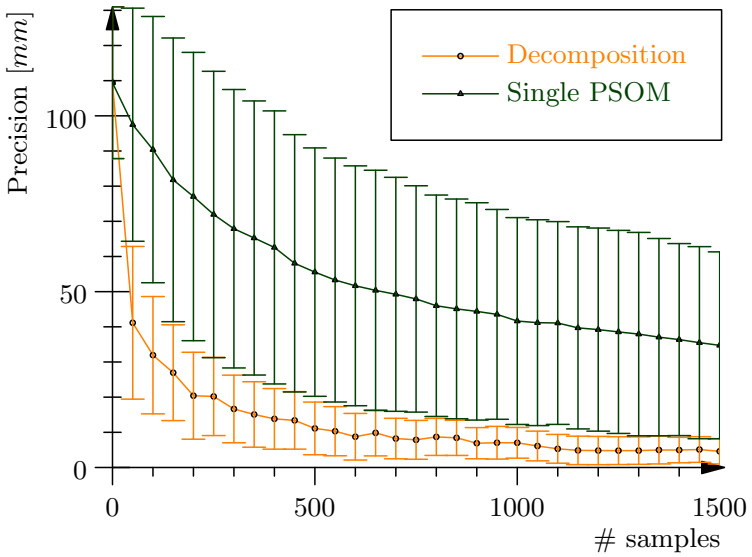
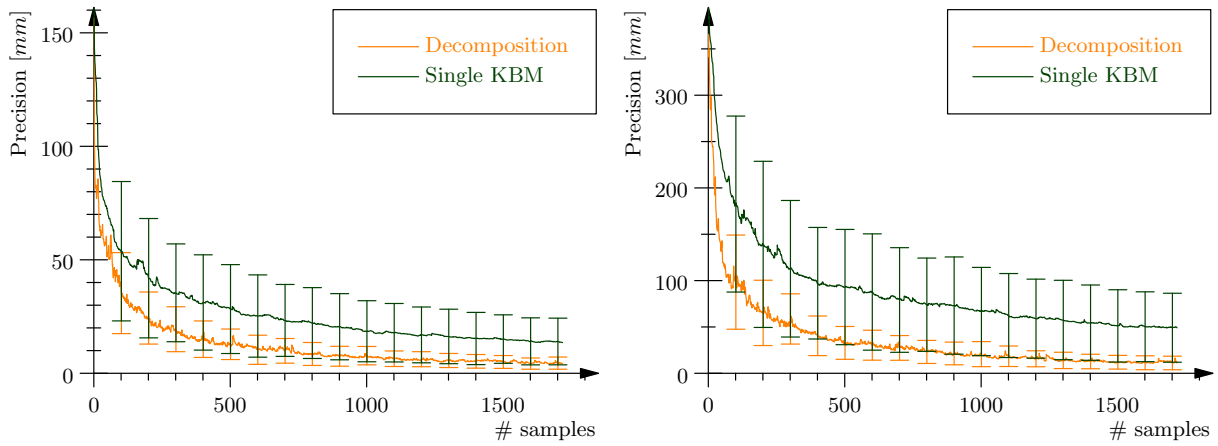


Figure 6.16.: Simulated shift in the joint angles absorbed with both networks learning simultaneously.

The last experiment with the PSOM simulates a shift in the joint angles. As this deformation affects the complete kinematics, both networks have to be trained simultaneously in analogy to the experiment above. Again, the results (see Figure 6.16) show that the decomposition absorbs the error much more quickly than a single network.

For the final experiments in this section, the incremental learning is applied to a decomposition learned with KBM models. As no noise is simulated, the models perfectly represent the kinematics before the deformation. Learning KBM models significantly requires less training samples compared to the incremental PSOM learning. For that reason, the simulated robot consists again of eight arm elements with independent degrees of freedom. The learn rate used for the individual KBM models is set to one and no mini-batch learning is applied in this experiment. The results displayed in Figure 6.17 confirm that the incremental learning of the KBM can equally be applied in a decomposition for both scenarios of tool use and shift in the joint angles.



(a) Simulated shift of 10° in each angle.

(b) Simulated tool use with an 400mm offset to end-effector.

Figure 6.17.: Experiments applying incremental learning to a decomposition with two KBM models on a simulated robot with eight degrees of freedom. Both models are trained simultaneously in both experiments.

7. Evaluation on a humanoid robot

In this chapter, the evaluation on a humanoid robot—the ARMAR-IIIb (Asfour et al., 2006)—is presented. For the experiments, three different data have been collected that were collected under different conditions. In addition to the data collected by the sensors available to the robot, the movements of the robot have been recorded with an external marker-based motion capture system providing high-precision observations which serve as ground truth data in the experiments (see Figure 7.2). For this setup, an additional software tool chain had to be developed to record and process the additional data. The components of this software are briefly presented in the Appendix B.1 and B.2.
Based on the experience gained from the previous section, two learning algorithms—learning with *partial least squares* and *incremental learning*—which are best suited for the application under realistic conditions were chosen for the evaluation with real data. In addition, the performance of the algorithm are compared to another state-of-the-art method for sensorimotor learning—namely the *locally weighted projection regression* (LWPR, see Section 2.3.4).

The first algorithm is the *partial least squares regression* (PLS-1, see Section 4.2) for offline learning because of its high robustness to sensor noise and the reduced tendency to overfitting even when learning from reduced training sets. The existence of few hyper parameters are also an important criterion for the application on a real system. The single hyper parameter of the PLS-1, however, directly reflects the statistics of the observed data and can be easily estimated especially when ground truth data is available.
The second algorithm that was applied is the incremental learning with the δ-rule (see Section 4.3). This learning has many advantages. First, it is very easy to implement and has a comparatively low computational demand which qualifies it for online learning. More importantly, it can be applied to refine previously learned models. If a kinematic model of the robot is available (which is the case for the ARMAR-III), it can be learned with zero prediction error in simulation and later be adapted to the real robot by the incremental learning.
The results of the experiments on the humanoid robot are presented in tabular form in the respective sections of this chapter. They are subsumed in the diagram in Figure 7.1 for an overview.

The following section describes how the data for the experiments was collected and how the data from the robot's sensors compare to the ground truth. After that, the experiments with the PLS-1 are presented. The third section presents the performance of the incremental learning that has been initialized with the kinematic model of the ARMAR-III.

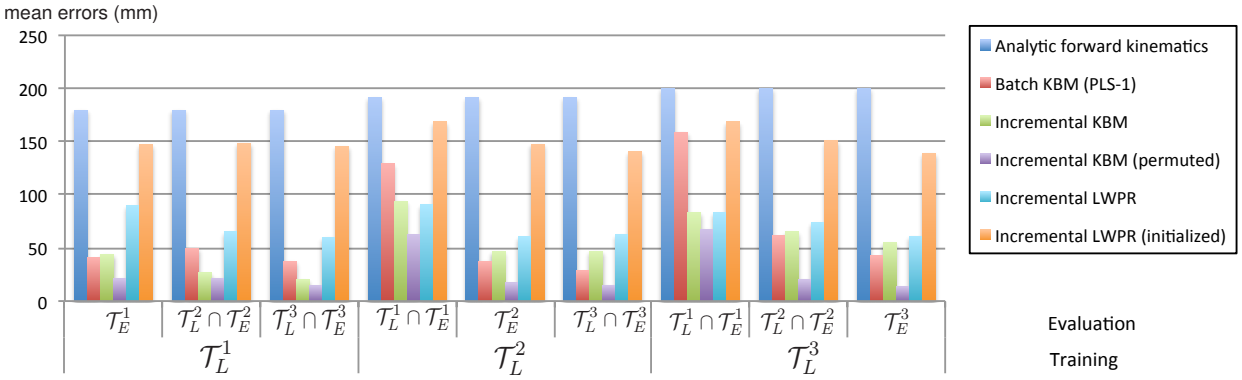


Figure 7.1.: Summary of the results of the experiments on the humanoid robot presented in this chapter.

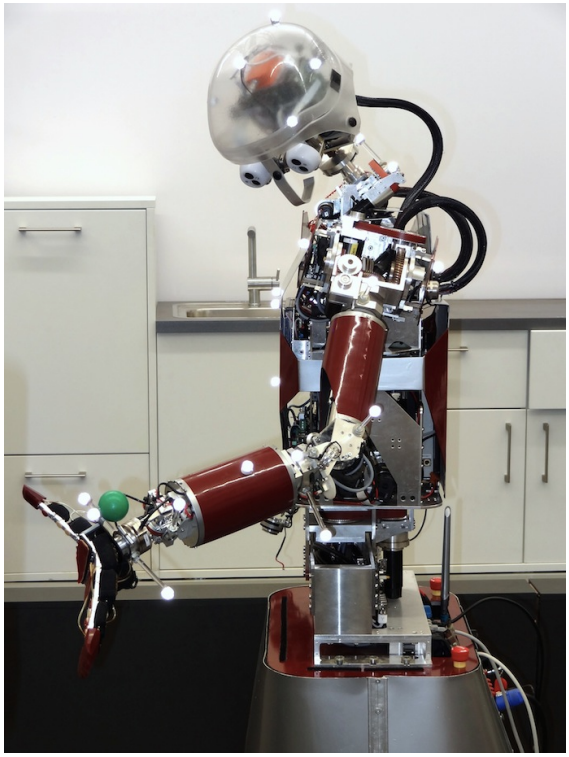


Figure 7.2.: The humanoid robot ARMAR-IIIb with optical markers attached to its left arm and head for motion capture.

7.1. Data acquisition

7.1.1. Setup

The robot platform used in the experiments in this chapter is the humanoid ARMAR-IIIb (Asfour et al., 2006) and is shown in Figure 7.2. It has an anthropomorphic upper body mounted on a mobile platform. Its arms has seven independent rotational degrees of freedom and pneumatic hands are attached to its wrists. The joints are either driven by V-belts (the shoulder joints) or tendons (elbow and wrist joints). The joint

positions are detected by incremental joint encoders only.
The head has four degrees of freedom in the neck and the eyes can be moved around individual pitch axes and a common tilt axis. In each eye, there are two cameras for foveal and peripheral vision. A spherical marker is attached to the wrist such that the spatial position of the hand can be detected by stereo vision. For the experiments, additional markers were attached that reflect the infrared light that is actively emitted by the cameras of a high-precision motion capture system. They are placed on each rigid body of the left arm as well as on the head, neck and torso.

The sensorimotor maps applied in the experiments learn the relation between the joint angles of the left arm and the position of the hand in the coordinate system of the torso. During data acquisition, the head was programmed to always look directly at the hand. For this reason, the joint angles of the head are not independent from the arm configuration and had to be excluded in the learning process. As the joints of the neck are not included in the kinematic chain to be learned, the inbuilt kinematics model of the head is used to transform the detected coordinates of the wrist into the coordinate frame of the torso. Although the utilization of a static kinematic model further increases the error of the stereo vision, the succeeding analysis reveals that the accuracy of the vision is in fact acceptable. On the one hand, this is because the robot always looks directly at the hand supporting the stereo reconstruction and, on the other hand, because the joints in the head are not tendon-driven. The joints in the elbow and wrist require repair more frequently causing a shift in the detected joint angles. In addition, the colored marker mounted on the wrist cannot not be exactly positioned in the kinematic model.

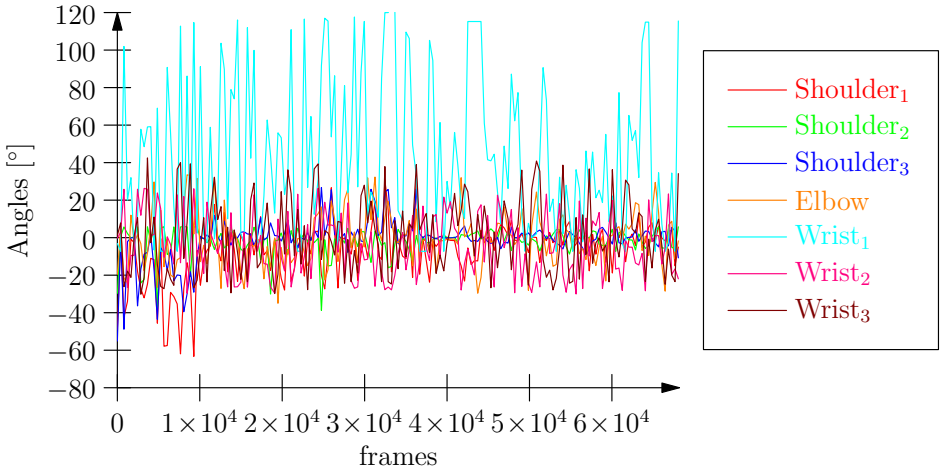
The ground truth data is generated by recording the exact positions of a marker attached to the wrist and at least three markers on the torso. Wrist positions are transformed into a coordinate frame in the torso defined by its three marker positions. Then the optimal rigid body transformation between the coordinate frames of the ground truth data and the vision is determined globally over the collected data following a method proposed by Besl and McKay (1992).

7.1.2. Data sets

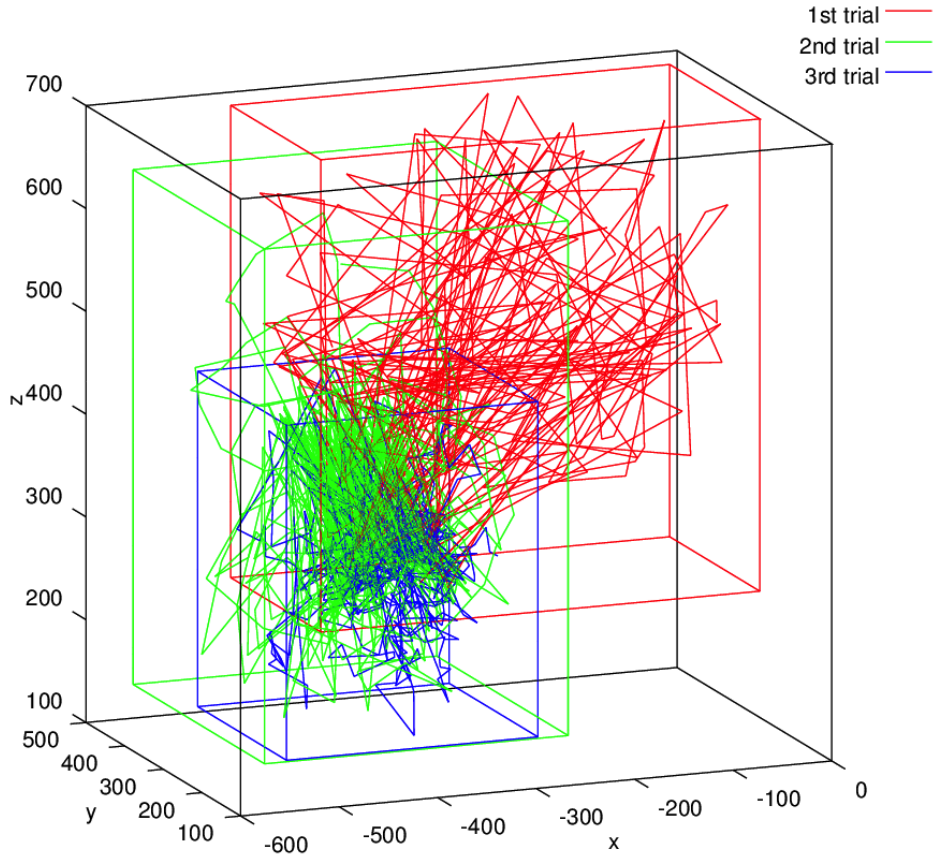
The data used in the experiments was recorded during three different trials under different conditions. Recording a sufficiently representative data set on robot with seven degrees of freedom is a difficult task. While the previous experiments conducted in simulation used uniformly distributed data, the high number of required movements rendered this intractable on the real robot. The data had to be recorded from random trajectories instead and has consequently a higher degree of redundancy which is less beneficial for learning. Even with a very high number of training samples, the algorithms run the risk of overfitting if the data is too similar. Figure 7.3 shows plots of the complete recorded data of the joint angles of the left arm.

In total, 66,944 samples were collected in the three trials leading to the complete data set

$$\mathscr{T} = \left\{ (\boldsymbol{\theta}_i, \boldsymbol{x}_i) : \boldsymbol{\theta}_i \in \Theta^7, \boldsymbol{x} \in \mathbb{R}^3, i \in \{1, \ldots, 66944\} \right\}. \tag{7.1}$$



(a) Recorded joint angles of all trials plotted after another.



(b) 3D plot of the trajectories of the end-effector. The colors indicate the different data sets recorded during each trial (no separation into sets for learning and evaluation is shown). Additionally, the bounding boxes are plotted to show the spatial expansion in the workspace.

Figure 7.3.: The data that has been collected on ARMAR-IIIb for the experiments.

For the experiments, it had been divided into the data sets of the individual trials which were then again divided into data sets for learning and evaluation. Therefore, always the first 5,000 samples were assigned for learning and the remaining samples for evaluation.
The first trial with 9,907 samples is the smallest contribution to the volume of the data set. However, the trajectories of this trial cover a big part of the configuration space and the relatively small number of samples is the result of a low sampling rate. The recorded data of this trial is split into two data sets $\mathscr{T}_L^1$ and $\mathscr{T}_E^1$ for learning and evaluation respectively:

$$\begin{aligned} \mathscr{T}_L^1 &= \Big\{(\boldsymbol{\theta}_i, \boldsymbol{x}_i) \in \mathscr{T} : i \in \{1, \ldots, 5000\}\Big\}, \qquad |\mathscr{T}_L^1| = 5000 \\ \mathscr{T}_E^1 &= \Big\{(\boldsymbol{\theta}_i, \boldsymbol{x}_i) \in \mathscr{T} : i \in \{5001, \ldots, 9907\}\Big\}, \quad |\mathscr{T}_E^1| = 4907 \\ \mathscr{T}^1 &= \mathscr{T}_L^1 \cap \mathscr{T}_E^1 \end{aligned} \tag{7.2}$$

The second trial created the biggest data sets with 31,684 samples. The trajectories were performed in a different area of the Cartesian space. Again the data is split up into data for learning and for evaluation:

$$\begin{aligned} \mathscr{T}_L^2 &= \Big\{(\boldsymbol{\theta}_i, \boldsymbol{x}_i) \in \mathscr{T} : i \in \{9908, \ldots, 14007\}\Big\}, \quad |\mathscr{T}_L^2| = 5000 \\ \mathscr{T}_E^2 &= \Big\{(\boldsymbol{\theta}_i, \boldsymbol{x}_i) \in \mathscr{T} : i \in \{14008, \ldots, 41591\}\Big\}, \quad |\mathscr{T}_E^2| = 26684 \\ \mathscr{T}^2 &= \mathscr{T}_L^1 \cap \mathscr{T}_E^2 \end{aligned} \tag{7.3}$$

The 22,353 samples observed during the last trial were recorded with a lower frame rate in a region close to that of the second trial. The resulting data sets are

$$\begin{aligned} \mathscr{T}_L^3 &= \Big\{(\boldsymbol{\theta}_i, \boldsymbol{x}_i) \in \mathscr{T} : i \in \{41592, \ldots, 46591\}\Big\}, \quad |\mathscr{T}_L^3| = 5000 \\ \mathscr{T}_E^3 &= \Big\{(\boldsymbol{\theta}_i, \boldsymbol{x}_i) \in \mathscr{T} : i \in \{46592, \ldots, 66944\}\Big\}, \quad |\mathscr{T}_E^3| = 20353. \\ \mathscr{T}^3 &= \mathscr{T}_L^1 \cap \mathscr{T}_E^3 \end{aligned} \tag{7.4}$$

7.1.3. Analysis of the data

This paragraph compares the data recorded by the robot's sensors to the ground truth data acquired by motion capture. Figure 7.4 displays a representative extract of the data showing the cartesian coordinates of the end-effector as recorded by the internal sensors (labeled as *'vision'*) and the motion capture (labeled as *'vicon'*) as well as the prediction of the inbuilt kinematic model of the robot (labeled as *'FK'*). The inaccuracy of the prediction of the kinematic model can be observed directly. However, the differences between the data of the motion capture data and the internal vision are surprisingly small. The highest error is in the x-coordinate. The errors introduced by stereo reconstruction is highest along the depth axis of the camera. By the transformation in the coordinate system of the torso the main part of this error is mapped onto the x-axis. Table. 7.1 shows the relative errors of the internal sensors and the kinematic model compared to the ground truth including the mean and median error, standard deviation ('std'), interquartile range ('iqr') and the max error ('max')—all errors are given in millimeters. For the vision, the errors are

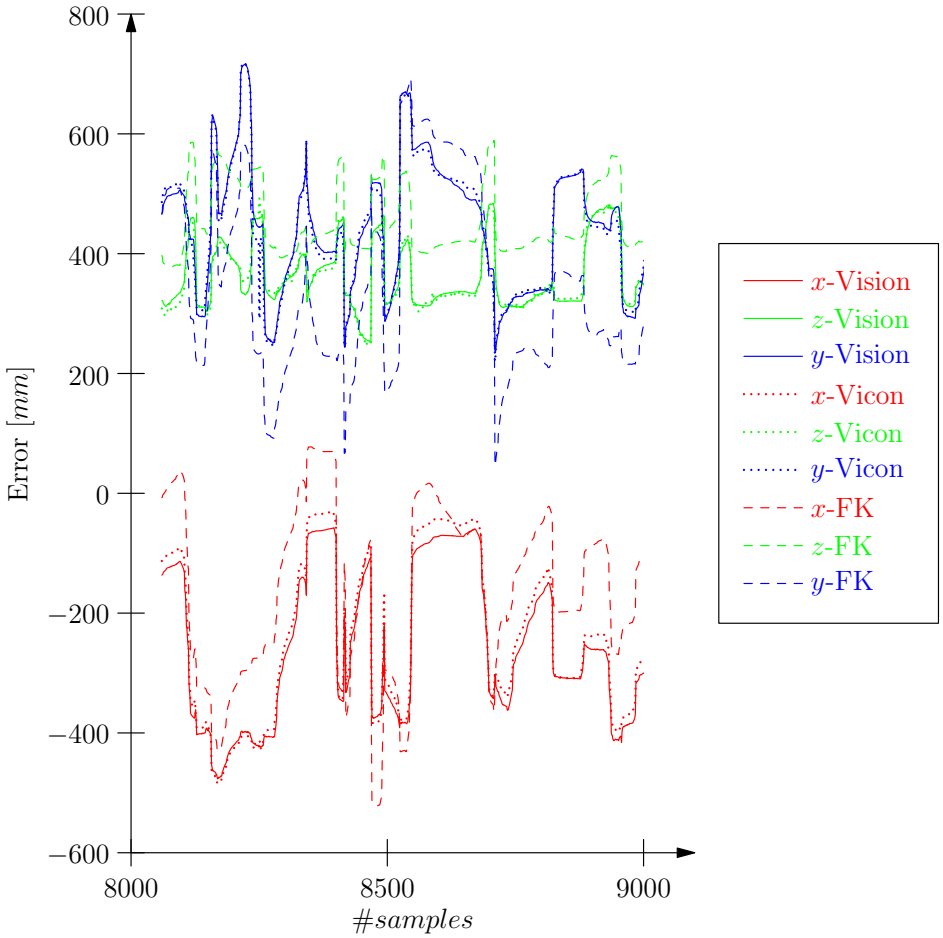


Figure 7.4.: Representative excerpt of the recorded spatial data. Three positions were recorded in each frame of the trajectories: The position detected by stereo vision *('vision')*, recorded by the motion capture *('vicon')* and the prediction by the inbuilt forward kinematics *('FK')*.

calculated for each coordinate individually as these statistics help to set the hyper parameter of the PLS-1 algorithm.

Data source	mean	median	std	iqr	max
Vision	14.7	12.1	9.8	10.3	163.5
x-coordinate	9.9	7.7	8.3	10.3	141.8
y-coordinate	5.8	4.3	5.4	5.7	136.0
z-coordinate	6.1	4.6	6.7	5.9	160.4
Kinematic model	190.7	191.1	32.7	51.4	270.4

Table 7.1.

The software developed to capture the motion of the robot (see Appendix B.2) uses the kinematic model of the ARMAR-IIIb to facilitate finding the correspondence of the observed marker positions and the virtual markers included in the model. Then the robot is moved to fit the observed marker positions to the virtual ones using the numerical inverse kinematics. Consequently, the software can also be used to estimate the

real joint configurations. It is however not its main purpose and the position of the markers in the model do not precisely coincide the real positions. For this reason, the estimated joint angles are not very precise and allow only a rough analysis of the recorded data. Especially the estimation for the lateral joints (for instance, the joints in the wrist) are very inaccurate. Still some conclusion can be drawn when comparing the values. Figure 7.5 shows the recorded data and the estimates (labeled with the suffix *'-IK'*). It can be seen that the more lateral the joint axis is, the higher the difference there is between the curves. The most interesting observation is that the values of the elbow joint are shifted by a constant offset of approximately seven degrees. This error is not unlikely as the tendons that drive the elbow joint are the most strained in the whole system. Every time the tendons have to be replaced, a shift in the encoder values is likely. As the elbow joint still has a strong influence on the end-effector position, this error can be held responsible for the majority of the errors of the prediction of the forward kinematics in Figure 7.4 and Table. 7.1.

7.1.4. Evaluation of partial least squares

Evaluation		Learning		
Test data	Errors	$\mathcal{T}^1$	$\mathcal{T}^2$	$\mathcal{T}^3$
$\mathcal{T}_L^1 \cap \mathcal{T}_E^1$	mean	41.1*)	129.1	158.7
	median	28.9*)	129.1	154.2
	iqr	27.3 *)	131.3	142.6
	std	43.8 *)	82.3	95.7
	max	342.0 *)	394.1 9	439.7 0
$\mathcal{T}_L^2 \cap \mathcal{T}_E^2$	mean	49.5	37.6*)	61.6
	median	35.1	24.9*)	47.3
	iqr	43.2	30.5*)	55.8
	std	44.3	36.1*)	48.6
	max	378.0	238.6*)	290.8 0
$\mathcal{T}_L^3 \cap \mathcal{T}_E^3$	mean	37.2	28.7	42.6*)
	median	33.1	25.5	36.5*)
	iqr	25.2	21.2	30.8*)
	std	20.4	16.7	29.3 *)
	max	127.6	149.0	167.5 *)

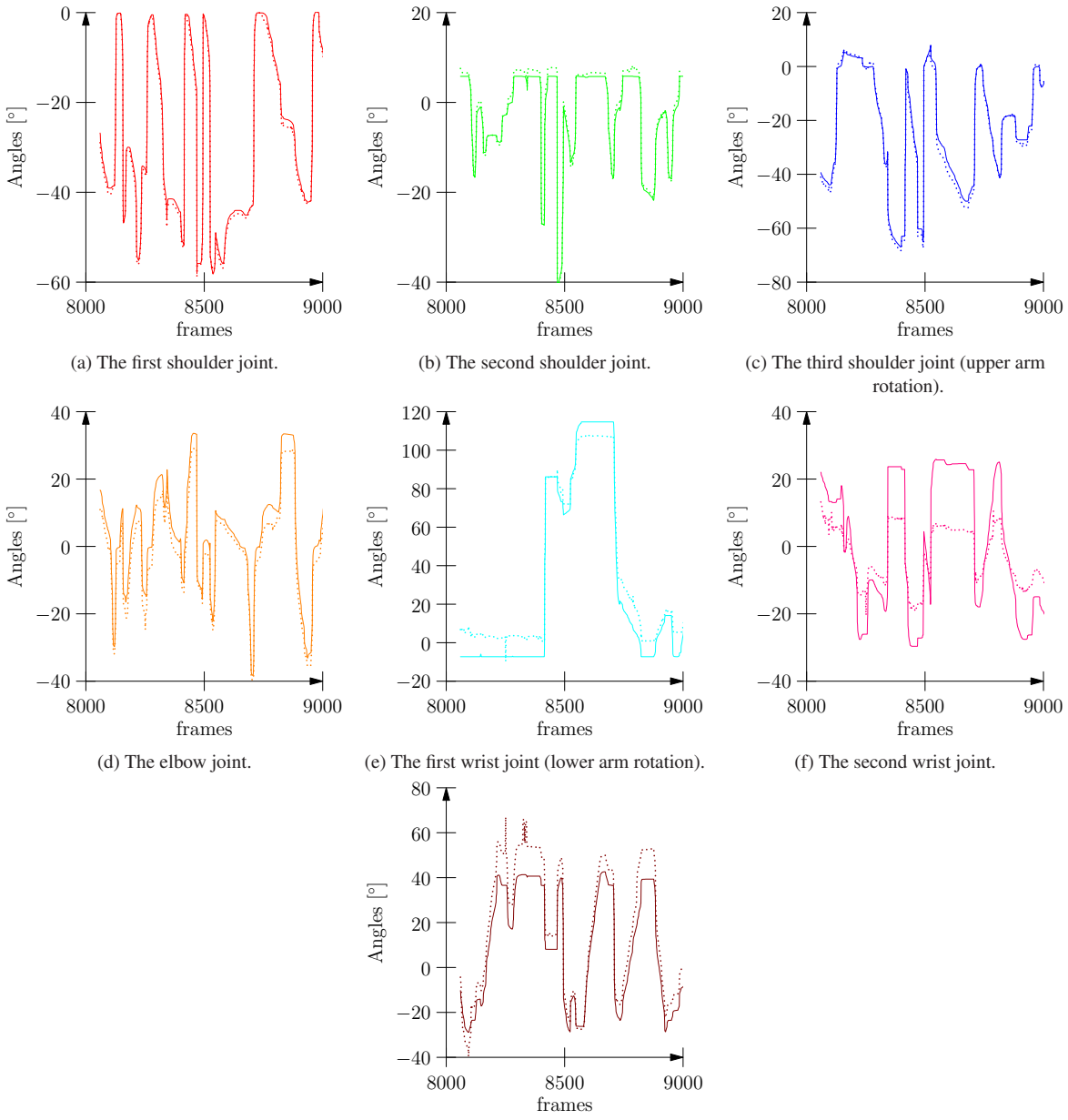
*) Models trained with $\mathcal{T}_L^i$ are evaluated only over $\mathcal{T}_E^i$ to avoid mixing training and test data.

Table 7.2.: Experiment with PLS-1 and the recorded data on the ARMAR-IIIb.

This paragraph describes the experiments performed with the real data collected during the experiments as described in the provious section. The experiments conducted follow the same pattern. At first, a training data set $\mathcal{T}_L^i$ is chosen and the KBM model is trained on its contained data. Then the model is evaluated over the respective evaluation set $\mathcal{T}_E^i$. Afterwards, the model is evaluated over the remaining data

$$\mathcal{T}_L^j \cap \mathcal{T}_E^j, \qquad j \neq i.$$

That way, it is guaranteed that the data sets for learning and evaluation remain separated in each experiment. The experiment is repeated for the remaining training sets $\mathcal{T}_L^j$.



(a) The first shoulder joint.

(b) The second shoulder joint.

(c) The third shoulder joint (upper arm rotation).

(d) The elbow joint.

(e) The first wrist joint (lower arm rotation).

(f) The second wrist joint.

(g) The third wrist joint.

Figure 7.5.: Validation of the joint encoders of the ARMAR-IIIb in the left arm. Continuous lines indicate the recorded joint values while dotted lines indicate reconstructed joint angles from the tracking software.

For the evaluation several statistics of the prediction errors over all sets used for evaluation are separately examined and presented in a tabular form: The mean and median errors (labeled as *'mean'* and *'median'*), the standard deviation (labeled as *'std'*), the interquartile range (the width of the interval that covers 50% of the values, labeled as *'iqr'*) and the maximal error (labeled as *'max'*). All errors are given in millimeters. The median and interquartile range are measures that are more tolerant to outliers and more meaningful as the prediction errors are not normally distributed. The results are shown in Table. 7.2.
The errors reflect how representative the data sets for learning are . When learning from $\mathscr{T}_L^1$, the errors over the evaluation sets are approximately equal with an acceptable mean precision of 4–5 cm and only a small number of outliers (see the values for standard deviation and interquartile range). The second data set is weaker. The errors over the first trial are significantly higher. However, the greater similarity of the third set is observable and manifests in low errors over this data set. The least representative data set was recorded in the third trial $\mathscr{T}_L^3$. The error over the evaluation data of the same trial $\mathscr{T}_E^3$ is larger compared to previous results and and the prediction is inaccurate especially over the first data set.
This analysis reveals that batch learning with KBM relies heavily on the data that is learned from. The pure number of training samples is therefore less important as their distribution. However, even under unfortunate conditions, the prediction error is not significantly worse than the prediction of the inbuilt kinematic model (see Tab. 7.1).

The same experiment was repeated using the LWPR for learning. Therefore, the freely available reference implementation has been used and the numerous hyper parameters were adjusted as suggested in the documentation. The results of this experiment are shown in Table. 7.3. In addition to the previous results, the number of receptive fields (labeled as 'RF') shown as well as the number of joint configurations that were not covered by any linear model and no prediction could be made for (labelled as 'Misses'). It can be seen that independently of the training set $\mathscr{T}_L^i$, the number of misses over the evaluation set of the same trial $\mathscr{T}_E^i$ is very high with about 20–25%. When learning with the data of the second and third trial, it is nearly impossible to make any predictions for the joint configurations of the first trial (a miss rate up to 78%). This can be explained by the observation that the first trial covers a completely different part of the work and configuration space and the learned model needs to extrapolate (see Fig. 7.3) where local learning methods do not excel at extrapolation by nature.
Even when a prediction is possible, the errors are significantly higher compared to the results of learning with the KBM.

7.2. Evaluation of the initialized incremental learning

The second series of experiments applies the incremental learning with the δ-rule. Incremental learning is capable of refining previously learned models that were created from an inaccurate forward model for instance. In these experiments, the inbuilt forward kinematics model of the ARMAR-IIIb is transformed into a KBM model by batch learning with the standard LLMS. Therefore, the same joint configurations θ_i out of the training set $\mathscr{T}_L^i$ are used and only the $\boldsymbol{x}_i$ are replaced by the predictions of the forward kinematics at the given joint configurations. The model then is an exact representation of the forward kinematics and

Evaluation		Learning $\mathcal{T}_L^1$	$\mathcal{T}_L^2$	$\mathcal{T}_L^3$
	# RFs	~ 329	~ 138	~ 44
Test data	Errors			
$\mathcal{T}_L^1 \cap \mathcal{T}_E^1$	Misses	26.40%*)	68.63%	78.1%
	mean	89.7*)	90.6	82.9
	std	77.1*)	78.1	63.6
	iqr	81.7*)	91.8	76.9
	median	61.7*)	61.7	54.8
	max	359.9*)	322.2	270.4
$\mathcal{T}_L^2 \cap \mathcal{T}_E^2$	Misses	27.80%	23.23%*)	36.6%
	mean	64.9	60.4*)	74.2
	std	55	52.4 *)	63.5
	iqr	52.2	55.5*)	62.8
	median	42.2	40.2*)	46.0
	max	281.3	231.5*)	258.8
$\mathcal{T}_L^3 \cap \mathcal{T}_E^3$	Misses	19.01%	13.94%	21.1%*)
	mean	60	62.5	61.0*)
	std	53.1	56.1	57.3*)
	iqr	40.6	46.4	40.1*)
	median	34.8	34.2	29.7*)
	max	237.4	198.9	200.8*)

*) Models trained with $\mathcal{T}_L^i$ are evaluated only over $\mathcal{T}_E^i$ to avoid mixing training and test data.

Table 7.3.: Experiment with LWPR and the recorded data on the ARMAR-IIIb.

can be adapted to the real hardware with incremental learning. Apart from the replacement of the learning algorithm, the experiments are conducted under the same conditions as in the previous experiments with batch learning.

In the first experiment, the data is processed by the algorithm in sequential order (that is, in the order they were recorded). This simulates online during the application of the robot where the samples can only be processed in this order. As a consequence, there is little difference between succeeding samples. To speed up learning, the training data is repeatedly presented to the algorithm. The results are shown in Table. 7.4 and the errors of the forward model over the individual data sets prior to refinement is given in the last column of the table. These outcomes confirm that the inclusion of a previously obtained approximative model greatly facilitates learning. Compared to Table. 7.2, the initialized incremental learning clearly outperforms the batch learning due to its greater embedded prior knowledge.

In the next experiment, it is investigated how much the order the training data is processed influences the quality of the prediction. The order of the data, in which it is processed, is therefore permuted. That way

Evaluation		Learning			
Test Data	Errors	$\mathcal{T}_L^1$	$\mathcal{T}_L^2$	$\mathcal{T}_L^3$	FK
$\mathcal{T}_L^1 \cap \mathcal{T}_E^1$	mean	44.1*)	94.9	83.3	179.6
	median	33.7*)	89.4	77.1	178.1
	iqr	38.1*)	77.9	58.5	55.7
	std	36.3*)	54.1	43.5	37.6
	max	216.2*)	380.5	219.5	280.5
$\mathcal{T}_L^2 \cap \mathcal{T}_E^2$	mean	26.6	46.8*)	65.1	191.5
	median	22.0	39.6*)	53.1	190.2
	iqr	16.0	35.8*)	53.0	47.3
	std	20.8	29.6*)	41.9	31.2
	max	178.1	202.9*)	203.3	265.8
$\mathcal{T}_L^3 \cap \mathcal{T}_E^3$	mean	20.0	47.0	55.5*)	200.4
	median	17.9	40.0	46.7*)	201.6
	iqr	14.2	32.9	47.9*)	57.3
	std	10.6	29.6	36.7*)	33.9
	max	74.0	220.6	180.2*)	269.5

*) Models trained with $\mathcal{T}_L^i$ are evaluated only over $\mathcal{T}_E^i$ to avoid mixing training and test data.

Table 7.4.: Incremental learning initialized by the outdated, manually crafted kinematic model of the ARMAR-III.

successive observations differ significantly which has a positive effect on the adaptation. Table. 7.5 reveals the remarkable results of the experiment. When learning from the first (the most representative) data set $\mathcal{T}_L^1$, the mean prediction errors are close to or even below. The values are already very close to the error of the ground truth data shown in Table. 7.1. Even when learned from the data set $\mathcal{T}_L^3$ (which is to be considered the least representative), the mean prediction errors remain below seven centimeters over all evaluation sets.

The approach of improving the learning by using an initialized model was also applied to the LWPR approach. Being a local learner, building a mapping that is valid for the whole configuration space is not feasible. For this experiment, the end effector positions to the joint configurations encountered in *all* data sets are calculated by the inbuilt forward kinematics of the ARMAR-IIIb. The LWPR learn then the positions at the joint configurations.
After the initialization, the experiment of the previous section is repeated. The final results are displayed in Table. 7.6. Now the miss rate over the evaluation data is below 1%. This time, the error over the data used for learning is shown as well in the first rows. The prediction errors over the training data is of comparable quality as the prediction errors of the learned KBM models over the evaluation data of the same trial. This suggests that the KBM and the LWPR produce comparable results for joint configurations similar to those during training.
The quality of the prediction of the learned LWPR model over the evaluation data is worse. This is due to the fact that each time the model could not make a prediction without the initialization, it now predicts a position close to the prediction of the inbuilt forward kinematics. For that reason, the errors are closer to the

Evaluation		Learning		
Test Data	Errors	$\mathcal{T}_L^1$	$\mathcal{T}_L^2$	$\mathcal{T}_L^3$
$\mathcal{T}_L^1 \cap \mathcal{T}_E^1$	mean	21.0	62.5	67.7
	median	15.9*)	51.5	58.0
	iqr	15.5*)	66.6	70.9
	std	20.8*)	46.9	52.0
	max	162.4*)	245.7	267.2
$\mathcal{T}_L^2 \cap \mathcal{T}_E^2$	mean	21.6*)	17.1	20.2
	median	16.0	12.0*)	15.1
	iqr	14.0	14.1*)	15.9
	std	20.5	15.6*)	17.5
	max	191.3	184.4*)	162.7
$\mathcal{T}_L^3 \cap \mathcal{T}_E^3$	mean	14.5	14.9	13.7*)
	median	12.0	12.9	11.9*)
	iqr	10.4	11.3	9.8*)
	std	9.5	9.3	8.4*)
	max	62.8	64.2	58.1*)

*) Models trained with $\mathcal{T}_L^i$ are evaluated only over $\mathcal{T}_E^i$ to avoid mixing training and test data.

Table 7.5.: Repetition of the experiment in Table. 7.4 with shuffled training data.

values in the lower row of Table. 7.1.

7.3. Conclusion

This chapter evaluated the batch learning of KBM models based on the partial least squares algorithm (PLS-1) and incremental learning after initialization by an inbuilt kinematics model.

The experiments showed that batch learning with PLS-1 is very robust to sensor noise and overfitting, and learns from sparse data. It is able to create fairly accurate models from scratch that allow prediction for unexplored regions of the configuration space. The initialized incremental learning, however, is more efficient and leads to better results. On most robot platforms, such an inbuilt, manually crafted model is available. Otherwise or if enough representative data can be collected, learning with PLS-1 is preferable.

Further, the two methods were compared to the LWPR algorithm. They both clearly outperform learning with LWPR under the evaluated conditions. This is not surprising as LWPR is a local learning method that shows its strength when learning lower-dimensional subspaces of the configuration space such as trajectories.

Evaluation Test data		Learning $\mathscr{T}_L^1$	$\mathscr{T}_L^2$	$\mathscr{T}_L^3$
	# RFs	$\sim$ 2306	$\sim$2239	$\sim$2143
Training data	Misses	0.3%	0.7%	0.3%
	mean	33.9	15.2	23.7
	std	22.4	10.0	15.6
	iqr	39.2	12.7	19.7
	median	30.7	16.8	26.0
	max	169.9	131.6	190.8
$\mathscr{T}_L^1 \cap \mathscr{T}_E^1$	Misses	0.1%*)	0.1%	0.1%
	mean	147*)	168.7	168.5
	std	149.9*)	170.0	168.0
	iqr	68*)	65.6	64.7
	median	54.7*)	55.2	53.6
	max	307.9*)	373.0	373.0
$\mathscr{T}_L^2 \cap \mathscr{T}_E^2$	Misses	0.4%	0.2%*)	0.3%
	mean	148.3	147.7*)	150.7
	std	152.9	153.9*)	159.1
	iqr	75.5	80.9*)	78.4
	median	56.5	58.3*)	56.9
	max	381.4	309.6*)	325.0
$\mathscr{T}_L^3 \cap \mathscr{T}_E^3$	Misses	0.3%	0.1%	0.2%*)
	mean	145.4	140.5	139.1*)
	std	147.4	144.2	142.5*)
	iqr	85.6	85.5	103.4*)
	median	62.3	62.3	64.6*)
	max	379.0	382.5	329.3*)

*) Models trained with $\mathscr{T}_L^i$ are evaluated only over $\mathscr{T}_E^i$ to avoid mixing training and test data.

Table 7.6.: Experiment with LWPR and the recorded data on the ARMAR-IIIb embedding prior knowledge.

8. Conclusion

The objective of this thesis was the development of novel methods that allow the implementation of artificial body schemas on humanoid robots inspired by the human model and that overcome the limitations of current approaches. For this purpose, the focus of this work lied on the development of novel model representations for sensorimotor maps and associated machine learning algorithms as well as on concepts for optimization by reducing the dimensionality of the problem. In this chapter, the contributions of this work are summarized, and ideas for extension and future work are discussed.

8.1. Contributions

An artificial body schema has to integrate signals from multiple sensor sources and motor signals androgen reflect their causal relationships. Such sensorimotor relations are of a complex and nonlinear nature and the high number of input dimensions impedes the construction of mappings between related sensor spaces. Learning in high-dimensional sensor spaces consequently is expensive. To learn accurate mappings, a large number of training examples has to be generated by sampling the possible combinations of signals. The number of samples required to form a precise mapping increases exponentially with the number of input dimensions. A common approach to overcome this problem is to create locally valid linear models that represent all configurations necessary to execute a limited set of expected actions with the robot. While linear models can be created rapidly and from few observations the resulting sensorimotor maps are limited to configurations similar to those explored during training, that is, they cannot extrapolate. Current approaches with global models represent sensorimotor relations over the complete configuration space. On the one hand, they require by far more data for their construction and are therefore not applicable on humanoid robots where the input dimensionality is usually very high. On the other hand, global models are required to control the robot also at joint configurations not previously sampled.
Global models that can efficiently be constructed from reasonable numbers of observations are therefore a prerequisite for the implementation of an artificial body schema on humanoid robots. For this reason, the focus of this thesis lied on the development of novel model representations for kinematic and dynamic sensorimotor maps, and techniques for dimensionality reduction that decrease the number of observations required for learning the kinematics. The methods have extensively been evaluated in simulation and on a real humanoid robot.
The key contributions of this work are listed below:

- **Models for sensorimotor maps**

The novel models of sensorimotor maps developed in this thesis are called *kinematic Bézier maps (KBM)* (Ulbrich et al., 2009, 2012c). While the state-of-the-art exclusively relies on models for general function regression, the KBM impose a more selective bias to learning. Being derived from projective geometry, they are specialized to the representation of combinations of trigonometric functions. Consequently, the models are restricted to represent direct kinematic and inverse dynamic functions (Ulbrich et al., 2012a). The reduced generality implies that the models can be derived from significantly lower numbers of observations compared to their more general counterparts. Another important property of the KBM is that each wmodel is linear in its parameters. This enables the application of very efficient machine learning algorithms and the construction of exact models in the absence of sensor noise. Unlike most models for general function approximation, the KBM depend only on a single hyper parameter that has only little effect on the learning of the model.
The mathematical background and the derivation of the models were presented in Chapter 3 and evaluated together with the associated learning algorithms both in simulation and experiments on a real humanoid platform and compared to state-of-the-art learning approaches.

- **Learning the sensorimotor models**
  The linearity of the model parameters in the KBM representation allowed for the adaptation and development of very efficient learning algorithms:

  i.) *Linear least squares methods* were the first to be investigated in this work. In the absence of sensor noise, the calculated models are an exact representation of the respective sensorimotor map. With noise, however, the standard approach is prone to overfitting and requires additional observation for compensation. For this reason, the partial least squares regression has been adapted for learning the KBM. By modeling the sensor noise, it becomes largely robust to overfitting and learns from sparse and noisy data. Experiments on a real humanoid platform revealed that this approach is the best choice when no additional prior knowledge on the robot is available (an approximative model for instance).

  ii.) *Incremental learning based on the δ-rule* allows the construction and refinement by successively processing single observations. The experiments on the real platform revealed that it is the most effective approach when prior knowledge in form of an iinbuilt model of the kinematics is available that can be updated.

  iii.) *nonlinear symmetry optimization* was the last learning algorithm that has been developed and evaluated in this work. It creates and maintains a symmetry in the model parameters of the KBM while minimizing the mean squared error over the training data. While the experimental results showed that it creates the most accurate models under all evaluated conditions, its slow computation, the sensibility to initialization, and the additional hyper parameter impede its application on real systems.

- **Dimensionality reduction through decomposition**
  Although the KBM representation greatly accelerates learning of sensorimotor maps, learning still re-

quires a number of training observations that is exponential in the number of input dimensions. In the case of learning kinematics, the decomposition proposed in this thesis (Ulbrich et al., 2012b) is able to significantly reduce this number independent of the machine learning method applied. The decomposition induces an exploration strategy that allows learning partial kinematic chains depending on fewer input variables. Unlike previous approaches, the proposed decomposition neither is limited to robots fulfilling structural constraints nor requires additional external sensors. Comparative experiments in simulation with different state-of-the-art learning methods and the KBM confirmed the efficiency of the proposed method.

8.2. Outlook

In this work, new methods for the representation and learning algorithms for sensorimotor maps have been presented that enable the efficient implementation of an artificial body schema on complex humanoid robots. This section proposes several ideas on how this representation can contribute to future research.

The KBM can model sensorimotor maps for the direct kinematics and inverse dynamics of a humanoid robot. As such, they either map joint angles to the position and/or orientation of the end-effector or a different body part, or map generalized coordinates (that is, joint positions, velocities and accelerations) to joint torques. While this is sufficient on the platforms used for evaluation, this approach cannot directly be applied to control a robot with alternative actuators such artificial pneumatic muscles for instance. Directly learning the relation between their complex motor signals and the outputs can be considered a valuable extension of the KBM-based approach towards more biologically inspired robots. Analogously, such an extension would open the opportunity to include the friction into the learning of the inverse dynamics which is currently not possible.

Learning KBM models is supervised which means that the algorithms process pairs of input and output values during training. In order to mimic the mirror neuron system of the human brain, it is necessary to convert learning with the KBM into pure unsupervised learning by simultaneously processing the position of several body parts to resolve redundancy.

Finally, the decomposition as a method for dimensionality reduction works only for the direct kinematics in its current state. Dynamic models have significantly more model parameters compared to kinematic models. As learning the inverse dynamics consequently is slower, the development of a decomposition pattern that can be applied to exploit redundancies in the inverse dynamics would be beneficial for learning.

A. Naming conventions

The following list provides the nomenclature used in the equations that appear in this chapter.

$\boldsymbol{a}, \boldsymbol{x}$	*small bold letters:* Cartesian vectors / spatial coordinates,
$\mathtt{a}, \mathtt{x}$	*small hollow letters:* Cartesian homogenous coordinates,
θ, α	*small greek letters:* joint vectors,
$\boldsymbol{\theta}, \boldsymbol{\alpha}$	*small bold greek letters:* vector of joint angles,
A, B	*capital letters:* matrices
X, Y	and random variable,
$((\boldsymbol{\theta}, \boldsymbol{x})$	tuples,)
$k(\cdot), b(\cdot), n(\cdot)$	*small letters with (at least one) argument:* general functions (kinematics, and models) with spatial output,
$\iota(\cdot), \pi(\cdot)$	*small greek letters with argument:* transformations and projections,
$B(\cdot), P(\cdot), L(\cdot)$	are exceptions to this rule. They denote polynomial basis functions.
$\mathscr{A}, \mathscr{P}, \mathscr{I}, \mathscr{T}$	*calligraphic capital letters:* spaces and sets
$\mathbb{R}, \mathbb{N}$	with the exception of the real and natural numbers
$\mathscr{N}, \mathscr{G}P$	and random distributions.
$\Theta^3 \subset \mathbb{R}^3$	*capital greek letters with natural exponent*: Joint angle spaces. Given a set of vectors $\mathscr{X} = \{\boldsymbol{x}_i : i = 1, \ldots, n\}$ the j-th component of the i-th vector is accessed by $x_{i,j}$
i, j, n, m	*small latin letters*: mostly natural index variables,
$d_{\text{in}}, d_{\text{out}}$	: reserved for the input and output dimensions.

B. Software

Alongside with the algorithms developed in this thesis, software has been developed partially in the context of the *OpenGRASP* project (Leon et al., 2010). OpenGRASP aims at the the creation of a simulation environment that especially targets grasping and dexterous object manipulation. While this is not the central topic of this work, two applications have been created that enrich the simulation environment but—more importantly—also foster the data acquisition for the experiments to come later. OpenGRASP, rather than creating an own dynamics simulator and visualization from scratch, builds around *OpenRAVE* an open-source project that features real-time robot dynamic simulation and a concise framework for the display and manipulation of virtual robot models and environmental objects created by Rosen Diankov (2010). The two projects this thesis contributes to this environment, on the one hand, is a software for the 3D design of robot models with a scientific focus on robotics and, on the other hand, a model-based tracking application that interfaces an optical high-precision and marker-based tracking application that interfaces an optical marker-based high-precision motion capture system that uses the facilities provided by OpenRAVE and the models created in the robot design software. The software provides the complete tool chain required in the experiments. Starting with the creation of approximative geometric robot models and recording of high-precision ground truth data from the motion capture together with the sensor readings of the robot. This leads to experiments in a caroled environment where uncertainty such as noise and decalibrations can be quantized and the learning can be evaluated objectively.

B.1. The robot modeling tool

The tool developed in this work for the robot model creation is called the *OpenGRASP Robot Editor* and was developed as a part of the *OpenGRASP* project in the context of the *GRASP* project founded as an FP7 project by the European union. Following the philosophy of OpenGRASP, it relies on established and expandable open-source software rather than designing such a complex project from scratch. The 3D modeling software *Blender*[1] has been chosen for the reason of being open-source, very versatile and expandable through a scripting interface and with the support of a very large and active community. The Robot Editor exploits the scripting interface to transform the software into a full-fledged modeling tool for real-life robots allowing the design of kinematics and dynamics models following the conventions used in robotics. Every parameter can be specified by the developer in a designated interface (see Figure B.1). Sensors and visual markers can be attached to the virtual robots' bodies which facilitated the development of a tracking software for model-based motion capture. In addition, the Robot Editor implements a novel and industry approved file

[1] `www.blender.org`

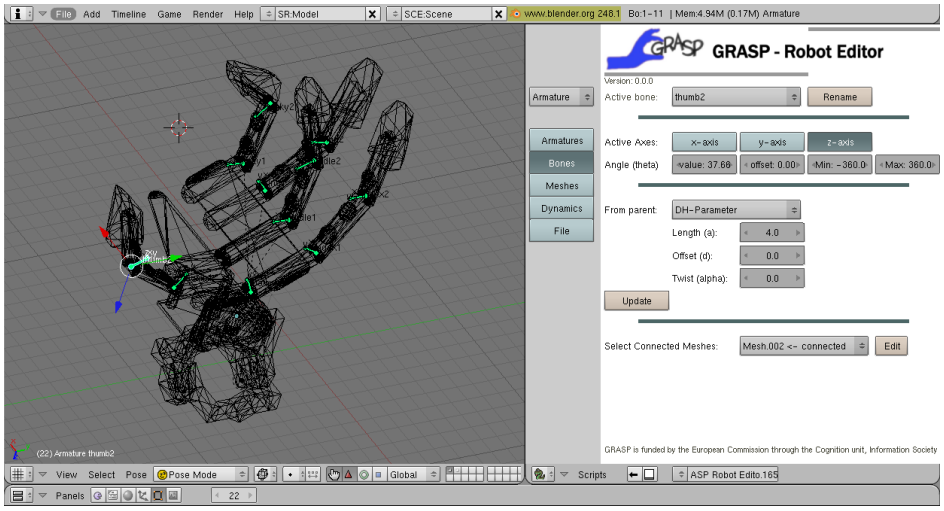
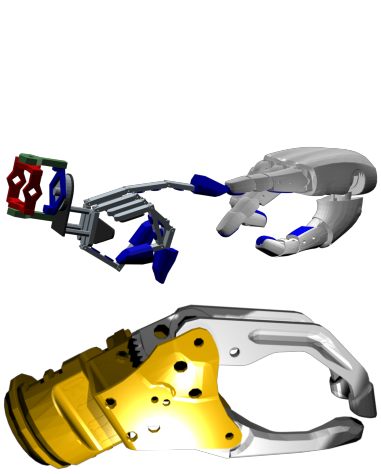
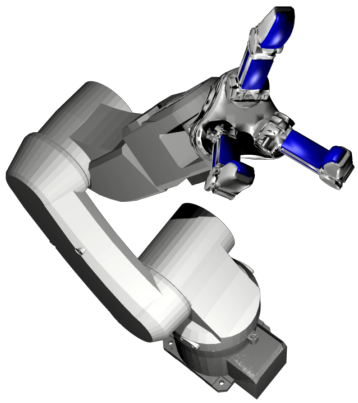


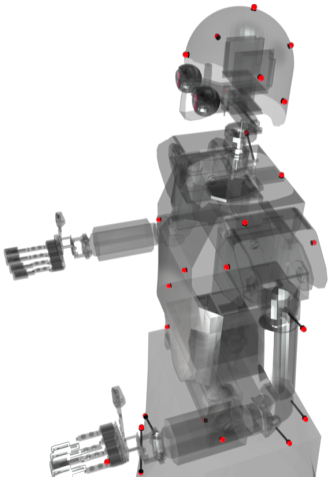
Figure B.1.: The designated user interface of the OpenGRASP Robot Editor for Blender.



(a) Assortment of robot hands (*ARMAR-III*, *Schunk SAH*, and the *DaVinci* prothesis by Otto Bock[a]).



(b) A *Schunk SHD hand* mounted on a *KUKA* manipulator.



(c) The *humanoid ARMAR-III* with attached optical marker positions.

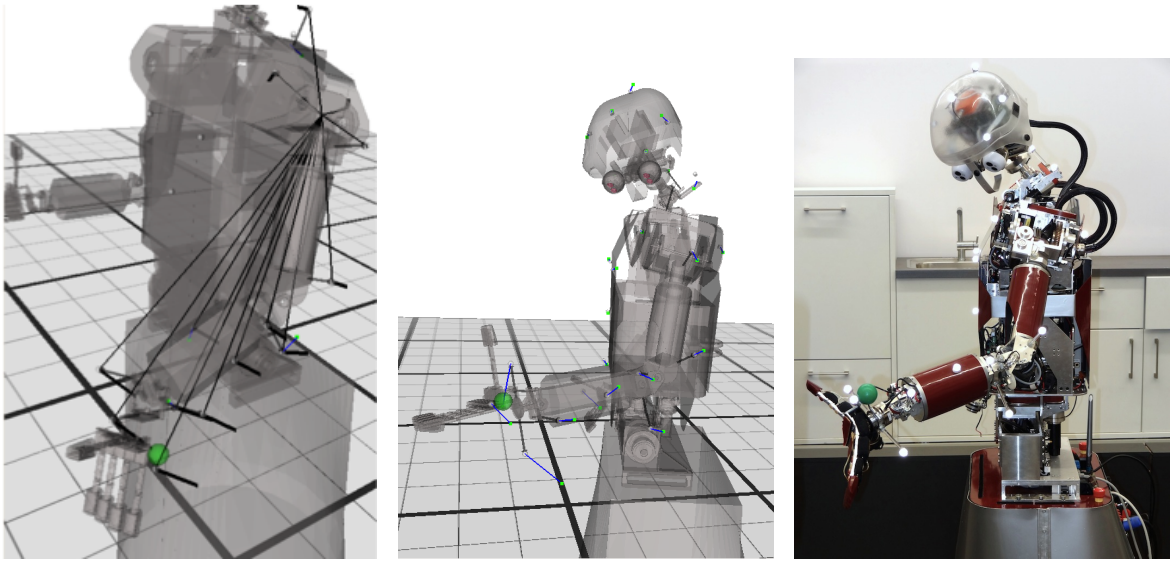
[a] `www.ottobock.com`

Figure B.2.: Models of manipulator, robots and hand models created with the robot editor.

format—*COLLADA* in version 1.5 (Barnes, 2006)—with a strong focus on kinematics and dynamics which is not natively supported by Blender and had been integrated into OpenRAVE during the development of the Robot Editor. It is largely backed by leading companies of the automotive and entertainment industries. Altogether, this makes the Robot Editor a a very useful tool completing a tool chain of robot simulation offered by OpenGRASP. This drastically reduces the effort for experimenting with virtual robots and robot hands in simulation.

Despite having been designed for modeling robot hands, it is not limited to that only and allows the creation of models of complete and complex models of humanoid robots and even of parametric models of humans for motion capture. On top of this, Blender enables the creation photo-realistic illustrations by ray tracing. A selection of hand models that were available in the GRASP consortium as well as a model of the humanoid robot ARMAR-III used in later experiments are displayed in Figure B.2 all created with the robot Editor.

B.2. The Tracking Studio



(a) The partial derivative of the forward kinematics with respect to the shoulder joint. Thin black lines indicate the distance to the joint axis and the thicker lines show the directions of the partial derivates.

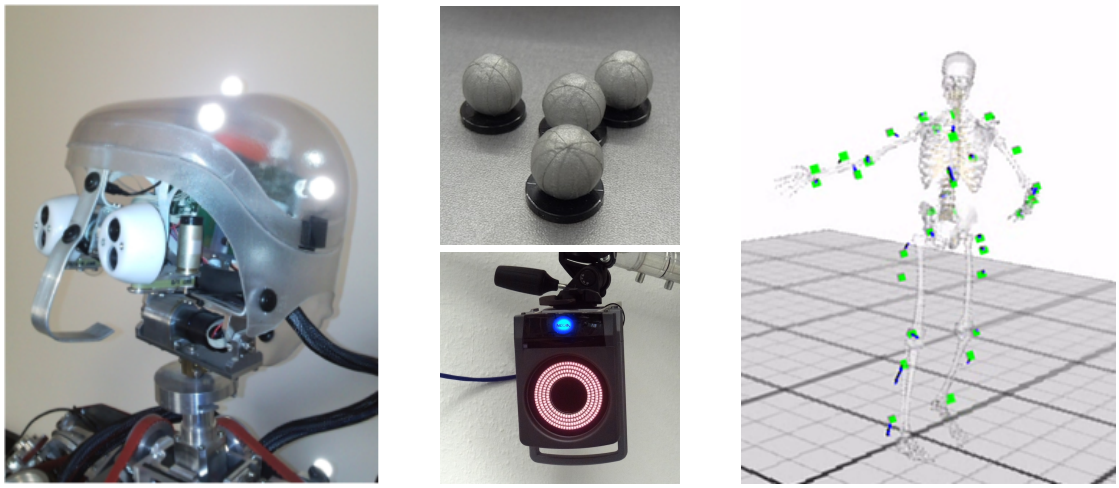
(b) Discrepancy between the real recorded data and the joint angles received by the robot. This shows the inaccuracy when applying an engineered model of the forward kinematics.

(c) The respective configuration of the real robot. The reflective optical markers and the green visual marker for the tracking of the end effector with the color-based stereo tracking system in the robot head.

Figure B.3.: Images of the TrackingStudio software developed in this work.

The evaluation of the learning algorithms is accompanied by many uncertainties mainly resulting from noise in joint decoders and the stereo vision system that is the most common technique which is also coherent with the paradigm of biological inspiration that excludes active sensors such as laser-scanner and pattern-projection. Of course, this uncertainty is the most important justification for the application of machine learning, apart from the difficulties of creating accurate robot models.

An objective and fair evaluation, however, requires quantization of these uncertainties. Motion capture equipment offers tracking of visual markers with an accuracy that qualifies it as ground truth data. However,



(a) Reflective markers attached to the head of the humanoid ARMAR-IIIb

(b) Active infrared camera and closeup of reflective markers.

(c) Snapshot of the TrackingStudio software for real-time human motion capture.

Figure B.4.: The components of the motion capture system.

motion capture in general and especially tracking of robots is not a trivial task that requires a lot of manual intervention. The software supplied with the motion capture system supports modeling joints and rigid bodies but one cannot benefit of the CAD model that is usually available or can be created with little effort with the software presented in the previous section. This is the reason why many markers have to be placed on each rigid body. In case of a human subject, this means on the upper body, head, upper and lower arms for instance. On a robot , rigid bodies between joints are very small such that many markers will lie in very close proximity which makes them difficult to track and distinguish, especially as the material of the robot is responsible for many reflections that further aggregates the situations.
Having a functional model greatly simplifies motion caputer as less markers are required (it is not even required to have on each rigid body) and the model restricts the possible motion and which simplifies the identification of markers and the tracking in general.
The simulation environment *OpenRAVE* features a scripting interface comparable to that available with Blender. In developing extensions to the simulator, it grants access to many useful tools such as visualization with annotation, mathematical tools and kinematics simulation. The motion capture system comes with a library for the integration into other applications such it can directly be accessed by the tracking extension.

All together, this resulted in the development of the *Tracking Studio*, a project that uses the OpenRAVE simulator as the foundation, interfaces the motion capture system and also communicate with the robot for collecting all sensor readings (see Figure B.4). The tracking algorithms that are applied are described in the following paragraphs.

Model-based tracking includes solving the inverse kinematics for all visible markers simultaneously once the correspondence between virtual markers and real positions found by the motion capture if initially found. The inverse kinematics moves the robot according to the recorded data. When markers are lost, for instance, due to overlapping, the model helps to find the matching marker becomes visible again. The inverse kinemat-

ics with so many model markers is overdetermined (see Figure B.3a) and, as there is only limited variation in between two subsequent frames, the kinematics can be linearly approximated. The Jacobian matrix, the matrix containing the partial derivatives given a joint configuration $\boldsymbol{\theta}$,

$$J(\boldsymbol{\theta}) = \begin{pmatrix} \frac{\partial \boldsymbol{x}_1}{\partial \theta_1} & \cdots & \frac{\partial \boldsymbol{x}_1}{\partial \theta_m} \\ \vdots & & \vdots \\ \frac{\partial \boldsymbol{x}_n}{\partial \theta_1} & \cdots & \frac{\partial \boldsymbol{x}_n}{\partial \theta_m} \end{pmatrix} \tag{B.1}$$

where n is the number of markers $\boldsymbol{x}_i$ that are visible at the time and m the number of joints of the robot model. In each frame, the linear approximation to the inverse kinematics

$$J(\boldsymbol{\theta}) \cdot \Delta = \Delta \boldsymbol{x} \tag{B.2}$$

is solved for $\Delta\boldsymbol{\theta}$ which is used to update the virtual robot model, and $\boldsymbol{x}$ contains the differences between the virtual model markers and the received real observed positions.

The virtual markers placed on the surface of the robot model cannot easily be positioned equally on the virtual robot model and the real robot. Further, the model itself is only an approximation to the robot kinematics (which is why learning is required in the first place). To lessen the effects resulting from this differences, the positions of virtual markers can be adjusted to the recorded positions online stabilizing the tracking enormously. In addition to the model-based tracking the TrackingStudio implements a nearest neighbor search in between subsequent frames and allows to assign the actual joint configuration received from the robot to the virtual model to find the initial starting position by the *iterative closes points* method.

Bibliography

K. C. Armel and V. S. Ramachandran. Projecting sensations to external objects: evidence from skin conductance response. *Proc. of the Royal Society of London. Series B: Biological Sciences*, 270(1523): 1499–1506, 2003.

M. Asada, K. F. MacDorman, H. Ishiguro, and Y. Kuniyoshi. Cognitive developmental robotics as a new paradigm for the design of humanoid robots. *Robotics and Autonomous Systems*, 37(2):185–193, 2001.

T. Asfour and R. Dillmann. Human-like motion of a humanoid robot arm based on closed-form solution of the inverse kinematics problem. In *IEEE/RSJ Intl. Conf. Intell. Robots, Syst. (IROS)*, pages 407–1412, 2003.

T. Asfour, K. Regenstein, P. Azad, J. Schröder, N. Vahrenkamp, and R. Dillmann. ARMAR-III: an integrated humanoid platform for sensory-motor control. In *IEEE/RAS Intl. Conf. Humanoid Robots*, page 169–175, Genova, Italy, Dec. 2006.

T. Asfour, J. Schill, H. Peters, C. Klas, J. Bücker, C. Sander, S. Schulz, A. Kargov, T. Werner, and V. Bartenbach. Armar-4: A 63 dof torque controlled humanoid robot. In *IEEE/RAS Intl. Conf. Humanoid Robots*, 2013.

M. Barnes. COLLADA. In *SIGGRAPH Courses*. ACM, 2006.

G. d. A. Barreto, A. F. R. Araújo, and H. J. Ritter. Self-organizing feature maps for modeling and control of robotic manipulators. *Journal of Intelligent and Robotic Systems*, 36(4):407–450, Apr. 2003.

L. Behera, M. Gopal, and S. Chaudhury. Self-organizing neural networks for learning inverse dynamics of robot manipulator. In *IEEE/IAS Intl. Conf. Industrial Automation and Control*, pages 457–460, 1995.

R. Bellman. Directions of mathematical research in nonlinear circuit theory. *IEEE Trans. Circuit Theory*, 7 (4):542–553, 1960.

D. J. Bennett, D. Geiger, and J. M. Hollerbach. Autonomous robot calibration for hand-eye coordination. *Intl. Journal of Robotics Research*, 10(5):550–559, Oct. 1991.

A. L. Benton. *Right-left discrimination and finger localization: development and pathology*. Hoeber-Harper, 1959.

G. Berlucchi and S. M. Aglioti. The body in the brain revisited. *Experimental Brain Research*, 200(1): 25–35, Aug. 2009.

N. A. Bernstein. *The co-ordination and regulation of movements*. Pergamon Press, 1967.

P. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(2):239–256, 1992.

C. M. Bishop. *Pattern recognition and machine learning*. springer New York, 2006.

M. Botvinick and J. Cohen. Rubber hands' feel'touch that eyes see. *Nature*, 391(6669):756–756, 1998.

V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology*. MIT Press, 1986.

S. Brown and C. Sammut. An architecture for tool use and learning in robots. In *Australasian Conf. Robotics and Automation*, 2007.

S. Brown and C. Sammut. Tool use and learning in robots. In N. Seel, editor, *Encyclopedia of the Sciences of Learning*, pages 3327–3330. Springer US, 2012.

M. V. Butz and O. Herbort. Context-dependent predictions and cognitive arm control with XCSF. In *Annual Conf. Genetic and evolutionary computation*, GECCO '08, page 1357–1364, New York, NY, USA, 2008. ACM.

M. V. Butz, G. K. Pedersen, and P. O. Stalph. Learning sensorimotor control structures with XCSF: redundancy exploitation and dynamic control. In *Proc. Annual Conf. on Genetic and evolutionary computation*, GECCO '09, page 1171–1178, New York, NY, USA, 2009. ACM.

M. Cabido-Lopes and J. Santos-Victor. Visual transformations in gesture imitation: What you see is what you do. In *IEEE Intl. Conf. Robot. Autom. (ICRA)*, volume 2, page 2375–2381, 2003.

S. Calinon, F. Guenter, and A. Billard. On learning, representing, and generalizing a task in a humanoid robot. *IEEE Trans. Syst., Man, Cybern. B*, 37(2):286–298, Apr. 2007.

E. Canzoneri, S. Ubaldi, V. Rastelli, A. Finisguerra, M. Bassolino, and A. Serino. Tool-use reshapes the boundaries of body and peripersonal space representations. *Experimental Brain Research*, 228(1):25–42, May 2013.

L. Cardinali, S. Jacobs, C. Brozzoli, F. Frassinetti, A. C. Roy, and A. Farnè. Grab an object with a tool and change your body: tool-use-dependent changes of body representation for action. *Experimental Brain Research*, 218(2):259–271, Feb. 2012.

S. Caselli, E. Faldella, B. Fringuelli, and L. Rosi. A neural approach to robotic haptic recognition of 3-d objects based on a kohonen self-organizing feature map. In *Intl. Conf. Industrial Electronics, Control and Instrumentation (IECON)*, volume 2, pages 835–840 vol.2, Sept. 1994.

S. P. Chatzis, D. Korkinof, and Y. Demiris. A nonparametric bayesian approach toward robot learning by demonstration. *Robot. Auton. Syst.*, 60(6):789–802, June 2012.

E. Chinellato, M. Antonelli, B. Grzyb, and A. del Pobil. Implicit sensorimotor mapping of the peripersonal space by gazing and reaching. *IEEE Trans. Autonomous Mental Development*, 3(1):43–53, Mar. 2011.

Y. Coiton, J. C. Gilhodes, J. L. Velay, and J. P. Roll. A neural network model for the intersensory coordination involved in goal-directed movements. *Biological cybernetics*, 66(2):167–176, 1991.

P. Corke. *Robotics, Vision and Control: Fundamental Algorithms in MATLAB*. Springer, Nov. 2011.

J. J. Craig. *Introduction to robotics: mechanics and control*. Pearson/Prentice Hall, Upper Saddle River, N.J., 2005.

H. Cruse, E. Wischmeyer, M. Brüwer, P. Brockfeld, and A. Dress. On the cost functions for the control of the human arm movement. *Biological cybernetics*, 62(6):519–528, 1990.

L. Csató and M. Opper. Sparse on-line gaussian processes. *Neural Computation*, 14(3):641–668, 2002.

B. Damas and M. Lopes. Robot manifolds for direct and inverse kinematics solutions. In *ROBOMAT 07*, 2007.

J. S. de la Cruz, W. Owen, and D. Kulic. Online learning of inverse dynamics via gaussian process regression. In *IEEE/RSJ Intl. Conf. Intell. Robots, Syst. (IROS)*, pages 3583–3590, Oct. 2012.

D. De Santis, V. Mohan, P. Morasso, and J. Zenzeri. Do humanoid robots need a body schema? In A. Chella, R. Pirrone, R. Sorbello, and K. R. Jóhannsdóttir, editors, *Biologically Inspired Cognitive Architectures 2012*, number 196 in Advances in Intelligent Systems and Computing, pages 109–115. Springer Berlin Heidelberg, Jan. 2013.

F. de Vignemont. Body schema and body image—Pros and cons. *Neuropsychologia*, 48(3):669–680, Feb. 2010.

D. Demers and K. Kreutz-Delgado. Canonical parameterization of excess motor degrees of freedom with self-organizing maps. *IEEE Trans. Neural Netw.*, 7(1):43–55, Jan. 1996.

R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.

N. Doidge. *The brain that changes itself: stories of personal triumph from the frontiers of brain science*. Viking, New York, 2007.

A. Droniou, S. Ivaldi, P. Stalph, M. Butz, and O. Sigaud. Learning velocity kinematics: Experimental comparison of on-line regression algorithms. *Proceedings robotica*, page 15–20, 2012.

A. D'Souza, S. Vijayakumar, and S. Schaal. Learning inverse kinematics. In *IEEE/RSJ Intl. Conf. Intell. Robots, Syst. (IROS)*, volume 1, page 298–303, 2001.

G. E. Farin. *NURBS: from projective geometry to practical use*. A.K. Peters, 1999.

A. G. Feldman. Functional tuning of the nervous system with control of movement or maintenance of a steady posture. *Biophysics*, 11:565–578, 1966.

M. Fiala. ARTag, a fiducial marker system using digital techniques. In *IEEE Computer Society Conf. Computer Vision, Pattern Recognition (CVPR)*, volume 2, pages 590–596 vol. 2, 2005.

P. Fitzpatrick and C. Metta. Grounding vision through experimental manipulation. *The Philosophical Transactions of the Royal Society: Mathematical, Physical, and Engineering Sciences*, 361:2165–2185, 2003.

F. Flentge. Locally weighted interpolating growing neural gas. *IEEE Trans. Neural Netw.*, 17(6):1382–1393, Nov. 2006.

l. E. Frank and J. H. Friedman. A statistical view of some chemometrics regression tools. *Technometrics*, 35(2):109–135, 1993.

S. Fuke, M. Ogino, and M. Asada. Body image constructed from motor and tactile images with visual information. *Intl. Journal of Humanoid Robotics*, 04(02):347–364, June 2007.

M. Fumagalli, A. Gijsberts, S. Ivaldi, L. Jamone, G. Metta, L. Natale, F. Nori, and G. Sandini. Learning to exploit proximal force sensing: A comparison approach. In O. Sigaud and J. Peters, editors, *From Motor Learning to Interaction Learning in Robots*, number 264 in Studies in Computational Intelligence, pages 149–167. Springer Berlin Heidelberg, Jan. 2010.

S. Gallagher and J. Cole. Body schema and body image in a deafferented subject. *Journal of Mind and Behavior*, 16(4):369–390, 1995.

C. Gaskett and G. Cheng. Online learning of a motor map for humanoid robot reaching. In *Intl. Conf. Computational Intelligence, Robotics and Autonomous Syst. (CIRAS)*, 2003.

J. Gerstmann. Problem of imperception of disease and of impaired body territories with organic lesions. *Archive of Neurology and Psychiatry*, 48:890–913, 1942.

Z. Ghahramani. Solving inverse problems using an EM approach to density estimation. In *Proc. Connectionist Models Summer School*, page 316–323, 1993.

A. Gijsberts and G. Metta. Incremental learning of robot dynamics using random features. In *IEEE Intl. Conf. Robot. Autom. (ICRA)*, page 951–956, 2011.

M. S. Graziano and C. G. Gross. A bimodal map of space: somatosensory receptive fields in the macaque putamen with corresponding visual receptive fields. *Experimental brain research. Experimentelle Hirnforschung. Expérimentation cérébrale*, 97(1):96–109, 1993.

P. Haggard and D. M. Wolpert. Disorders of body scheme. In *In Higher-Order Motor Disorders*. University Press, 2005.

C. Hartmann, J. Boedecker, O. Obst, S. Ikemoto, and M. Asada. Real-time inverse dynamics learning for musculoskeletal robots based on echo state gaussian process regression. In *Proc. of Robotics: Science and Systems*, Sydney, Australia, 2012.

H. Head and G. Holmes. Sensory disturbances from cerebral lesions. *Brain*, 34(2-3):102–254, Nov. 1911.

D. Hebb. *The organization of behavior*. Wiley, New York, 1949.

T. Heed and B. Röder. The body in a multisensory world. In M. M. Murray and M. T. Wallace, editors, *The Neural Bases of Multisensory Processes*, Frontiers in Neuroscience. CRC Press, Boca Raton (FL), 2012.

M. Hersch, E. Sauser, and A. Billard. Online learning of the body schema. *Intl. Journal of Humanoid Robotics*, 05(02):161–181, June 2008.

M. Hersch, A. Billard, and S. Bergmann. Iterative estimation of rigid-body transformations: Application to robust object tracking and iterative closest point. *Journal of Mathematical Imaging and Vision*, 43(1): 1–9, Apr. 2011.

T. Hesselroth, K. Sarkar, P. Van Der Smagt, and K. Schulten. Neural network control of a pneumatic robot arm. *IEEE Trans. Syst., Man, Cybern.*, 24(1):28–38, Jan. 1994.

M. Hikita, S. Fuke, M. Ogino, and M. Asada. Cross-modal body representation based on visual attention by saliency. In *IEEE/RSJ Intl. Conf. Intell. Robots, Syst. (IROS)*, pages 2041–2046, Sept. 2008.

H. Hoffmann, S. Schaal, and S. Vijayakumar. Local dimensionality reduction for non-parametric regression. *Neural processing letters*, 29(2):109–131, 2009.

M. Hoffmann, H. G. Marques, A. H. Arieta, H. Sumioka, M. Lungarella, and R. Pfeifer. Body schema in robotics: A review. *IEEE T. Autonomous Mental Development*, 2(4):304–324, 2010.

J. M. Hollerbach. A recursive lagrangian formulation of manipulator dynamics and a comparative study of dynamics formulation complexity. *IEEE Trans. Syst., Man, Cybern. B*, 10(11):730–736, Nov. 1980.

J. M. Hollerbach and C. W. Wampler. The calibration index and taxonomy for robot kinematic calibration methods. *The Intl. Journal of robotics research*, 15(6):573–591, 1996.

N. P. Holmes. Does tool use extend peripersonal space? a review and re-analysis. *Experimental Brain Research*, 218(2):273–282, Mar. 2012.

K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Netw.*, 2(5):359–366, July 1989.

A. Iriki, M. Tanaka, and Y. Iwamura. Coding of modified body schema during tool use by macaque post-central neurones. *Neuroreport*, 7(14):2325–2330, Oct. 1996.

A. Iriki, M. Tanaka, S. Obayashi, and Y. Iwamura. Self-images in the video monitor coded by monkey intraparietal neurons. *Neuroscience research*, 40(2):163–173, June 2001.

J. Jockusch and H. Ritter. An instantaneous topological mapping model for correlated stimuli. In *Intl. Joint Conf. Neural Networks (IJCNN)*, volume 1, pages 529–534, 1999.

M. Jones and D. Vernon. Using neural networks to learn hand-eye co-ordination. *Neural Computing & Applications*, 2(1):2–12, 1994.

M. I. Jordan and D. E. Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16(3):307–354, 1992.

K. Kaneko, F. Kanehiro, M. Morisawa, K. Miura, S. Nakaoka, and S. Kajita. Cybernetic human HRP-4C. In *IEEE/RAS Intl. Conf. Humanoid Robots*, pages 7–14, 2009.

D. Katz, Y. Pyuro, and O. Brock. Learning to manipulate articulated objects in unstructured environments using a grounded relational representation. In *Proc. Robotics: Science and Systems IV*, pages 254–261, Zurich, Switzerland, June 2008.

H. Kihl, J. P. Urban, J. Gresser, and S. Hagmann. Neural network based hand-eye positioning with a transputer-based system. In B. Hertzberger and G. Serazzi, editors, *High-Performance Computing and Networking*, pages 281–286. Springer Berlin Heidelberg, Jan. 1995.

K. Kilteni, J.-M. Normand, M. V. Sanchez-Vives, and M. Slater. Extending body space in immersive virtual reality: A very long arm illusion. *PLoS ONE*, 7(7):e40867, July 2012.

S.-Y. King and J.-N. Hwang. Neural network architectures for robotic applications. *IEEE Trans. Robot. Autom.*, 5(5):641–657, 1989.

S. Klanke and H. Ritter. PSOM+: parametrized self-organizing maps for noisy and incomplete data. In *Proc. Workshop Self-Organizing Maps (WSOM)*, 2005.

Y. Kobayashi and S. Hosoe. Planning-space shift motion generation: Variable-space motion planning toward flexible extension of body schema. *Journal of Intelligent & Robotic Systems*, 62(3-4):467–500, Sept. 2010.

T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43(1): 59–69, Jan. 1982.

P. Kovács and G. Hommel. On the tangent-half-angle substitution. In J. Angeles, G. Hommel, and P. Kovács, editors, *Computational Kinematics*, number 28 in Solid Mechanics and Its Applications, pages 27–39. Springer Netherlands, Jan. 1993.

J. R. Lackner. Some proprioceptive influences on the perceptual representation of body shape and orientation. *Brain*, 111(2):281–297, Apr. 1988.

A. Lammert, L. Goldstein, S. Narayanan, and K. Iskarous. Statistical methods for estimation of direct and differential kinematics of the vocal tract. *Speech Communication*, 55(1):147–161, Jan. 2013.

B. Leon, S. Ulbrich, R. Diankov, G. Puche, M. Przybylski, A. Morales, T. Asfour, S. Moisio, J. Bohg, J. Kuffner, and R. Dillmann. OpenGRASP: a toolkit for robot grasping simulation. In *Intl. Conf. Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR)*, Darmstadt, Germany, Nov. 2010.

J. W. Lewis. Cortical networks related to human use of tools. *The Neuroscientist*, 12(3):211–231, June 2006.

L. Li, W. A. Gruver, Q. Zhang, and Z. Yang. Kinematic control of redundant robots and the motion optimizability measure. *IEEE Trans. Syst., Man, Cybern. B*, 31(1):155–160, 2001. PMID: 18244778.

A. Liegeois. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE Trans. Syst., Man, Cybern.*, 7(12):868–871, Dec. 1977.

L. Ljung. *System identification: theory for the user*. Prentice-Hall, Upper Saddle River, NJ, 2009.

L. Ljung and T. Söderström. *Theory and practice of recursive identification*. MIT Press series in signal processing, optimization, and control. MIT Press, 1985.

M. R. Longo, M. P. M. Kammers, H. Gomi, M. Tsakiris, and P. Haggard. Contraction of body representation induced by proprioceptive conflict. *Current biology: CB*, 19(17):R727–R728, Sept. 2009.

M. Lopes and B. Damas. A learning framework for generic sensory-motor maps. In *IEEE/RSJ Intl. Conf. Intell. Robots, Syst. (IROS)*, pages 1533–1538, Nov. 2007.

M. Lopes and J. Santos-Victor. Visual learning by imitation with motor representations. *IEEE Trans. Syst., Man, Cybern. B*, 35(3):438–449, 2005.

M. Lopes and J. Santos-Victor. Learning sensory-motor maps for redundant robots. In *IEEE/RSJ Intl. Conf. Intell. Robots, Syst. (IROS)*, page 2670–2676, 2006.

B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proc. Imaging Understanding Workshop*, pages 121–130, 1981.

D. Ma and J. Hollerbach. Identifying mass parameters for gravity compensation and automatic torque sensor calibration. In *IEEE Intl. Conf. Robot. Autom. (ICRA)*, volume 1, pages 661–666, Apr. 1996.

A. Maravita and A. Iriki. Tools for the body (schema). *Trends in cognitive sciences*, 8(2):79–86, Feb. 2004.

A. Maravita, C. Spence, and J. Driver. Multisensory integration and the body schema: close to hand and within reach. *Current Biology*, 13(13):R531–R539, July 2003.

T. Martinetz and K. Schulten. A "Neural-Gas" network learns topologies. *Artificial Neural Networks*, I: 397–402, 1991.

T. Martinetz and K. Schulten. A neural network for robot control: Cooperation between neural units as a requirement for learning. *CEE*, 19(4):315–332, 1993.

R. Martinez-Cantin, M. Lopes, and L. Montesano. Body schema acquisition through active learning. In *IEEE Intl. Conf. Robot. Autom. (ICRA)*, pages 1860–1866, 2010.

L. Massone. Sensorimotor learning. In *The Handbook of Brain Theory and Neural Networks*. MIT Press, 1994.

J.-P. Merlet. Interval analysis for certified numerical solution of problems in robotics. *Intl. Journal of Applied Mathematics and Computer Science*, 19(3):399–412, 2009.

G. Metta, G. Sandini, and J. Konczak. A developmental approach to visually-guided reaching in artificial systems. *Neural networks*, 12(10):1413–1427, 1999.

G. Metta, R. Manzotti, F. Panerai, and G. Sandini. Development: is it the right way towards humanoid robotics. In *Intl. Conf. Intell. Auton. Syst. (IAS)*, 2000.

G. Metta, G. Sandini, and L. Natale. Sensorimotor interaction in a developing robot. In *1st intl. Workshop on epigenetic robotics: Modeling cognitive development in robotic systems*, page 18–19. Lund University Press, 2001.

G. Metta, G. Sandini, D. Vernon, L. Natale, and F. Nori. The icub humanoid robot: an open platform for research in embodied cognition. In *Proc. the 8th Workshop on Performance Metrics for Intelligent Systems*, PerMIS '08, pages 50–56, New York, NY, USA, 2008. ACM.

V. Mohan and P. Morasso. Towards reasoning and coordinating action in the mental space. *Intl. Journal of Neural Systems*, 17(04):329–341, 2007.

V. Mohan and P. Morasso. How past experience, imitation and practice can be combined to swiftly learn to use novel "tools": Insights from skill learning experiments with baby humanoids. In *Biomimetic and Biohybrid Systems*, page 180–191. Springer, 2012.

V. Mohan, P. Morasso, G. Metta, and G. Sandini. A biomimetic, force-field based computational model for motion planning and bimanual coordination in humanoid robots. *Autonomous robots*, 27(3):291–307, 2009.

V. Mohan, P. Morasso, G. Metta, and S. Kasderidis. Actions and imagined actions in cognitive robots. In *Perception-Action Cycle*, page 539–572. Springer, 2011a.

V. Mohan, P. Morasso, J. Zenzeri, G. Metta, V. S. Chakravarthy, and G. Sandini. Teaching a humanoid robot to draw 'Shapes'. *Autonomous Robots*, 31(1):21–53, 2011b.

V. Mohan, P. Morasso, and G. Sandini. A neural frame work for organization and flexible utilization of episodic memory in 'cumulatively' learning baby humanoids. *Neural Comput*, 2013.

P. Morasso. What is the use of the body schema for humanoid robots? *Intl. Journal of Machine Consciousness*, 05(01):75–94, June 2013.

P. Morasso and V. Sanguineti. Self-organizing body schema for motor planning. *Journal of Motor Behavior*, 27(1):52–66, 1995.

A. Murata, L. Fadiga, L. Fogassi, V. Gallese, V. Raos, and G. Rizzolatti. Object representation in the ventral premotor cortex (area f5) of the monkey. *Journal of neurophysiology*, 78(4):2226–2230, 1997.

F. A. Mussa Ivaldi, P. Morasso, and R. Zaccaria. Kinematic networks. *Biological Cybernetics*, 60(1):1–16, Nov. 1988.

C. Nabeshima and Y. Kuniyoshi. A method for sustaining consistent sensory-motor coordination under body property changes including tool Grasp/Release. *Advanced Robotics*, pages 687–717, 2010.

C. Nabeshima, Y. Kuniyoshi, and M. Lungarella. Adaptive body schema for robotic tool-use. *Advanced Robotics*, 20(10):1105–1126, 2006.

C. Nabeshima, Y. Kuniyoshi, and M. Lungarella. Towards a model for tool-body assimilation and adaptive tool-use. In *IEEE Intl. Conf. Development and Learning (ICDL)*, pages 288–293, July 2007.

L. Natale. A developmental approach to grasping. In *In Developmental Robotics AAAI Spring Symposium*, 2005.

L. Natale, F. Nori, G. Sandini, and G. Metta. Learning precise 3D reaching in a humanoid robot. In *IEEE Intl. Conf. Development and Learning (ICDL)*, pages 324–329, July 2007.

D. Nguyen-Tuong and J. Peters. Model learning for robot control: a survey. *Cognitive Processing*, page 1–22, 2011.

D. Nguyen-Tuong, J. Peters, M. Seeger, and B. Schölkopf. Learning inverse dynamics: a comparison. In *16th European Symposium on Artificial Neural Networks*, pages 13–18, Bruges, Belgium, 2008.

D. Nguyen-Tuong, B. Scholkopf, and J. Peters. Sparse online model learning for robot control with support vector regression. In *IEEE/RSJ Intl. Conf. Intell. Robots, Syst. (IROS)*, pages 3121–3126, Oct. 2009.

C. Nolker and H. Ritter. Visual recognition of continuous hand postures. *IEEE Trans. Neural Netw.*, 13(4): 983–994, 2002.

F. Nori, L. Natale, G. Sandini, and G. Metta. Autonomous learning of 3d reaching in a humanoid robot. In *IEEE/RSJ Intl. Conf. Intell. Robots, Syst. (IROS)*, page 1142–1147, 2007.

E. Oztop, L.-H. Lin, M. Kawato, and G. Cheng. Dexterous skills transfer by extending human body schema to a robotic hand. In *IEEE/RAS Intl. Conf. Humanoid Robots*, pages 82–87, Dec. 2006.

J. Paillard. Body schema and body image – a double dissociation in deafferented patients. In G. Gantchev, S. Mori, and M. J., editors, *Motor Control: Today and Tomorrow*, pages 197–214. Academic Publishing House: Sofia, Bulgaria, 1999.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-Spline Techniques*. Springer, Oct. 2002.

C. E. Rasmussen and C. K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Nov. 2005.

C. L. Reed and M. J. Farah. The psychological reality of the body schema: a test with normal participants. *Journal of experimental psychology. Human perception and performance*, 21(2):334–343, Apr. 1995.

J. B. Ritchie and T. A. Carlson. Tool integration and dynamic touch. *Psychological Science*, 24(6):1066–1068, June 2013.

H. Ritter. Parameterized self-organizing maps. In *Proc. Intl. Conf. on Artificial Neural Networks*, pages 568–575, 1993.

H. Ritter, T. Martinetz, and K. Schulten. *Neural computation and self-organizing maps: an introduction*. Addison-Wesley, 1992.

H. J. Ritter, T. M. Martinetz, and K. J. Schulten. Topology-conserving maps for learning visuo-motor-coordination. *Neural Networks*, 2(3):159–168, 1989.

M. Rolf, J. Steil, and M. Gienger. Bootstrapping inverse kinematics with goal babbling. In *IEEE Intl. Conf. Development and Learning (ICDL)*, pages 147–154, Aug. 2010a.

M. Rolf, J. Steil, and M. Gienger. Goal babbling permits direct learning of inverse kinematics. *IEEE Trans. Autonomous Mental Development*, 2(3):216–229, Sept. 2010b.

M. Rolf, J. Steil, and M. Gienger. Learning flexible full body kinematics for humanoid tool use. In *Intl. Conf. Emerging Security Technologies (EST)*, pages 171–176, Sept. 2010c.

C. Rorden, J. Heutink, E. Greenfield, and I. H. Robertson. When a rubber hand'feels' what the real hand cannot. *Neuroreport*, 10(1):135–138, 1999.

V. Ruiz de Angulo and C. Torras. Self-calibration of a space robot. *IEEE Trans. Neural Netw.*, 8(4):951–963, 1997.

V. Ruiz de Angulo and C. Torras. Using PSOMs to learn inverse kinematics through virtual decomposition of the robot. In J. Cabestany, A. Prieto, and F. Sandoval, editors, *Computational Intelligence and Bioinspired Systems*, number 3512, pages 701–708. Springer Berlin Heidelberg, Jan. 2005a.

V. Ruiz de Angulo and C. Torras. Speeding up the learning of robot kinematics through function decomposition. *IEEE Trans. Neural Netw.*, 16(6):1504–1512, 2005b.

V. Ruiz de Angulo and C. Torras. Learning inverse kinematics: Reduced sampling through decomposition into virtual robots. *IEEE Trans. Syst., Man, Cybern. B*, 38(6):1571–1577, 2008.

C. Salaün, V. Padois, and O. Sigaud. Control of redundant robots using learned models: An operational space control approach. In *IEEE/RSJ Intl. Conf. Intell. Robots, Syst. (IROS)*, pages 878–885, Piscataway, NJ, USA, Oct. 2009.

S. Schaal and C. G. Atkeson. Memory-based robot learning. In *IEEE Intl. Conf. Robot. Autom. (ICRA)*, pages 2928–2933, 1994a.

S. Schaal and C. G. Atkeson. Robot juggling: implementation of memory-based learning. *Control Systems IEEE*, 14(1):57–71, 1994b.

S. Schaal and C. G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10:2047–2084, 1997.

S. Schaal, S. Vijayakumar, and C. G. Atkeson. Local dimensionality reduction. *Advances in neural information processing systems*, pages 633–639, 1998.

S. Schaal, C. G. Atkeson, and S. Vijayakumar. Real-time robot learning with locally weighted statistical learning. In *IEEE Intl. Conf. Robot. Autom. (ICRA)*, page 288–293, 2000.

S. Schaal, C. G. Atkeson, and S. Vijayakumar. Scalable techniques from nonparametric statistics for real time robot learning. *Applied Intelligence*, 17(1):49–60, June 2002.

L. Sciavicco and B. Siciliano. *Modelling and control of robot manipulators: with 190 figures*. Springer, Apr. 2000.

R. W. Shumaker, K. R. Walkup, and B. B. Beck. *Animal Tool Behavior: The Use and Manufacture of Tools by Animals*. The Johns Hopkins University Press, revised and updated edition, Apr. 2011.

G. Sicard, C. Salaün, S. Ivaldi, V. Padois, , and O. Sigaud. Learning the velocity kinematics of ICUB for model-based control: XCSF versus LWPR. In *IEEE/RAS Intl. Conf. Humanoid Robots*, pages 570–575, Oct. 2011.

O. Sigaud, C. Salaün, and V. Padois. On-line regression algorithms for learning mechanical models of robots: a survey. *Robotics and Autonomous Systems*, 59:1115–1129, 2011.

J. Sinapov and A. Stoytchev. Toward autonomous learning of an ontology of tool affordances by a robot. In *AAAI Conf. Artificial Intelligence*, pages 1828–1829, 2008.

J. F. Soechting and M. Flanders. Errors in pointing are due to approximations in sensorimotor transformations. *Journal of Neurophysiology*, 62(2):595–608, 1989.

P. O. Stalph, M. V. Butz, and G. K. Pedersen. Controlling a four degree of freedom arm in 3D using the XCSF learning classifier system. In *Advances in Artificial Intelligence KI*, page 193–200. Springer, 2009.

P. O. Stalph, J. Rubinsztajn, O. Sigaud, and M. V. Butz. A comparative study: function approximation with LWPR and XCSF. In *Annual Conf. Genetic and evolutionary computation*, GECCO '10, page 1863–1870, New York, NY, USA, 2010. ACM.

M. Stamenov. Body schema, body image, and mirror neurons. In H. De Preester and V. Knockaert, editors, *Body Image and Body Schema*, pages 21–43. Amsterdam and Philadelphia: John Benjamins, 2005.

A. Stoytchev. Computational model for an extendable robot body schema, 2003.

A. Stoytchev. *Robot tool behavior: A developmental approach to autonomous tool use*. PhD thesis, College of Computing, Georgia Institute of Technology, 2007.

A. Stoytchev. Learning the affordances of tools using a behavior-grounded approach. In E. Rome, J. Hertzberg, and G. Dorffner, editors, *Towards Affordance-Based Robot Control*, number 4760 in Lecture Notes in Computer Science, pages 140–158. Springer Berlin Heidelberg, Jan. 2008.

J. Sturm, C. Plagemann, and W. Burgard. Body schema learning for robotic manipulators from visual self-perception. *Journal of Physiology-Paris*, 103(3–5):220–231, 2009.

J. Sturm, C. Plagemann, and W. Burgard. Body schema learning. In E. Prassler, M. Zöllner, R. Bischoff, W. Burgard, R. Haschke, M. Hägele, G. Lawitzky, B. Nebel, P. Plöger, and U. Reiser, editors, *Towards Service Robots for Everyday Environments*, number 76 in Springer Tracts in Advanced Robotics, pages 131–161. Springer Berlin Heidelberg, Jan. 2012.

J. Sun de la Cruz, D. Kulic, and W. Owen. Learning inverse dynamics for redundant manipulator control. In *Intl. Conf. Autonomous and Intell. Syst. (AIS)*, page 1–6, 2010.

J. Sun de la Cruz, D. Kulic, and W. Owen. Online incremental learning of inverse dynamics incorporating prior knowledge. In M. Kamel, F. Karray, W. Gueaieb, and A. Khamis, editors, *Autonomous and Intelligent Systems*, number 6752, pages 167–176. Springer Berlin Heidelberg, Jan. 2011.

A. Tessari and R. I. Rumiati. Motor distal component and pragmatic representation of objects. *Brain research. Cognitive brain research*, 14(2):218–227, Aug. 2002.

S. Ulbrich and T. Asfour. Improving body schema learning with Kinematic Bézier Maps by symmetry constraints. In *Workshop Autonom. Learning*, IEEE Intl. Conf. Robot. Autom. (ICRA), Karlruhe, Germany, May 2013.

S. Ulbrich, V. Ruiz, T. Asfour, C. Torras, and R. Dillmann. Rapid learning of humanoid body schemas with Kinematic Bézier Maps. In *IEEE/RAS Intl. Conf. Humanoid Robots*, pages 431–438, Dec. 2009.

S. Ulbrich, M. Bechtel, T. Asfour, and R. Dillmann. Learning robot dynamics with Kinematic Bézier Maps. In *IEEE/RSJ Intl. Conf. Intell. Robots, Syst. (IROS)*, Vilamoura, Portugal, Oct. 2012a.

S. Ulbrich, V. Ruiz de Angulo, T. Asfour, C. Torras, and R. Dillmann. General robot kinematics decomposition without intermediate markers. *IEEE Trans. Neural Netw., Learning Syst.*, 23(4):620–630, Apr. 2012b.

S. Ulbrich, V. Ruiz de Angulo, T. Asfour, C. Torras, and R. Dillmann. Kinematic Bézier Maps. *IEEE Trans. Syst., Man, Cybern.*, 42(4):1215–1230, Aug. 2012c.

R. J. van Beers, D. M. Wolpert, and P. Haggard. When feeling is more important than seeing in sensorimotor adaptation. *Current biology: CB*, 12(10):834–837, May 2002.

D. Vasquez, T. Fraichard, and C. Laugier. Growing hidden markov models: An incremental tool for learning and predicting human and vehicle motion. *The Intl. Journal of Robotics Research*, 28(11-12):1486–1506, Nov. 2009.

S. Vijayakumar and S. Schaal. Locally weighted projection regression: An o (n) algorithm for incremental real time learning in high dimensional space. In *Intl. Conf. on Machine Learning (ICML)*, volume 1, page 288–293, 2000.

J. Walter. *Rapid Learning in Robotics*. PhD thesis, University of Bielefeld, Bielefeld, 1996.

J. Walter. PSOM network: learning with few examples. In *IEEE Intl. Conf. Robot. Autom. (ICRA)*, volume 3, pages 2054–2059, 1998.

J. Walter and H. Ritter. Rapid learning with parametrized self-organizing maps. *Neurocomputing*, 12: 131–153, 1995.

D. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Trans. Man-Machine Syst.*, 10(2):47–53, June 1969.

B. Widrow and M. E. Hoff. Adaptive switching circuits. *IRE WESCON Convention Record*, 4, 1960.

H. Wold. Soft modeling by latent variables; the nonlinear iterative partial least squares approach. *Perspectives in Probability and Statistics. Papers in Honour of M. S. Bartlett*, 1975.

D. M. Wolpert, R. C. Miall, and M. Kawato. Internal models in the cerebellum. *Trends in cognitive sciences*, 2(9):338–347, 1998.

Y. S. Xia, G. Feng, and J. Wang. A primal-dual neural network for online resolving constrained kinematic redundancy in robot motion control. *IEEE Trans. Syst., Man, Cybern. B*, 35(1):54–64, Feb. 2005. PMID: 15719933.

Y. Yoshikawa, H. Kawanishi, M. Asada, and K. Hosoda. Body scheme acquisition by cross modal map learning among tactile, visual, and proprioceptive spaces. In *Proc. Intl. Workshop on Epigenetic Robotics*, pages 181–184, 2002.

Y. Yoshikawa, Y. Tsuji, K. Hosoda, and M. Asada. Is it my body? body extraction from uninterpreted sensory data based on the invariance of multiple sensory attributes. In *IEEE/RSJ Intl. Conf. Intell. Robots, Syst. (IROS)*, volume 3, pages 2325–2330, Oct. 2004.

Y. Yoshikawa, K. Hosoda, and M. Asada. Unique association between self-occlusion and double-touching towards binding vision and touch. *Neurocomputing*, 70(13–15):2234–2244, Aug. 2007.

M. Zeller, R. Sharma, and K. Schulten. Motion planning of a pneumatic robot using a neural network. *IEEE Control Systems*, 17(3):89–98, June 1997.