Workflow-based Context-aware Control of Surgical Robots

Zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

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

genehmigte

Dissertation

von

Tim Beyl

aus Heilbronn

Tag der mündlichen Prüfung:	20. November 2015
Erster Gutachter:	Prof. Dr.-Ing. Heinz Wörn
Zweiter Gutachter:	Dr. Ferdinando Rodriguez y Baena

Dedication and Credits

This thesis was created in the time from 2011-2014 at the Institute for Anthropomatics and Robotics - Institute for Process Control and Robotics at the Karlsruhe Institute of Technology. Parts of the work have been created within the research projects *Active Constraints Technologies for Ill-defined or Volatile Environments* funded within the seventh framework program of the EU, *Single-Port-Technologie für gastroenterologische und viszeralchirurgische endoskopische Interventionen* funded by the German Research Foundation and *Kontextsensitive Assistenz im aufmerksamen OP* founded by the Federal Ministry of Education and Research (Germany).

Firstly, i would like to thank my supervisor Prof. Dr.-Ing. Dr. h.c. Wörn for offering the great opportunity to me to work in the field of surgical robotics, for his supervision, assistance and for pushing me in the right direction. Many thanks go to my second reviewer Dr. Ferdinando Rodriguez y Baena, who not only reviewed my thesis and with whom i had great and fruitful discussions about my work, but who also backed and supported my colleagues and me in the research project ACTIVE FP7. Another thank goes to my group leader Dr. rer. nat. Jörg Raczkowsky, who always was a great backer in every difficult situation, who challenged me with discussions about my work and who helped to significantly improved my work through his support during my time at the IPR.

Thanks go to all of my colleagues, especially to Thorsten Brennecke, Julien Mintenbeck, Philip Nicolai, Luzie Schreiter, Yaokun Zhang, Holger Mönnich, Gavin Kane and Markus Mehrwald from the MeGI group for their collaboration and the friendly atmosphere. A special thank goes to my students Christian Biegemeier, Stefan König, Steffen Slavetinsky, Thomas Bilich and Andreas Böhme who wrote their thesis under my guidance. Big thanks go to Mirko Kunze, with whom i worked closely together during the ACTIVE project and with whom i had fruitful discussions about our joint work, which improved the quality of my work.

I thank all my friends and club colleagues for making the rare free time more enjoyable.

I dedicate this work to my parents for supporting me for already nearly 30 years. Thank you for everything!

Out of clutter, find simplicity.
From discord, find harmony.
In the middle of difficulty lies opportunity.

Albert Einstein

Abstract

Surgical assistance system such as medical robots enhanced the capabilities of medical procedures in the last decades. Newer trends are the operating room integration and surgical phase recognition. These among others serve the purpose to optimize assistance systems and their use. This work presents a new perspective on the use of workflows in the operating room and shows methods to include not only the procedure and patient specific workflow into the process, but also to include the peri-operative workflow in the operating room and focuses on the medical personnel. This is accomplished by a 3D perception system that is able to track and to detect the personnel in the entire operating room that is described in this work. A workflow-based controller has been developed and evaluated, which is equipped with a knowledge base and that allows to interpret sensory information from the operating room to follow the actual workflow. The workflow in this scope can be designed by a domain expert (e.g. surgeon) using business process modelling and a graphical representation. The knowledge base is used during run time to find the optimal components for the operation and to configure them for every task included to the workflow. The perception system has been evaluated quantitatively. The workflow-based controller has been used to execute a workflow that combines autonomous, hands-on and telemanipulation actions of a surgical robot, which has been evaluated by performing user studies. The evaluation has been performed in a laboratory setup. The results show a high performance of the developed perception system and the feasibility as well as an excellent user experience of the workflow detection and control approach using the workflow-based controller

Zusammenfassung

Motivation

Computerassistierte Chirurgie erhält immer stärkeren Einzug in moderne Operationssääle. Systeme wie chirurgische Roboter, oder Navigationssysteme bieten erhebliches Potential für die Verbesserung der Eingriffsqualität und für die Unterstützung des chirurgischen Personals. Obgleich aktuelle Systeme beeindruckende Möglichkeiten eröffnen wird der medizinische Arbeitsablauf für die Steuerung solcher Systeme bisher nur in Ansätzen berücksichtigt. Gründe dafür ergeben sich aus dem zum Teil gravierenden Eingriff in den medizinischen Arbeitslauf beim Einsatz von chirurgischen Robotern, der Komplexität der Integration verschiedenster Teilsysteme, fehlender Kompatibilität und der aufwendigen Echtzeitverfolgung des medizinischen Arbeitsablaufes.

Ziel dieser Arbeit ist die Entwicklung, die prototypische Implementierung, sowie die Evaluierung von Methoden zur Workflow-basierten Steuerung von chirurgischen Robotern. Der Ansatz soll sowohl die einfache Modellierung von Arbeitsabläufen durch medizinisches Personal und die Verfolgung des Arbeitsablaufes zur Laufzeit, als auch die optimale Komposition und Konfiguration der Assistenzsysteme ermöglichen. Schwerpunkte werden auf die generische Behandlung von chirurgischen Aufgaben und der Komponenten des Assistenzsystemes gelegt. Dies ermöglicht eine aufgabenspezifisch optimale Konfiguration des Assistenzsystemes, einfache Anpassung der Arbeitsabläufe an den Patienten oder das Personal und die schnelle Integration neuer Komponenten in das Assistenzsystem. Das Assistenzsystem soll hierbei das Personal möglichst gut unterstützen, ohne die Aufmerksamkeit des Personals von der eigentlichen chirurgischen Aufgabe abzuziehen.

Für die Verfolgung des Arbeitsablaufes während der Laufzeit wird ein Umgebungsüberwachungssystem auf Basis von RGB-D Kameras entwickelt. Dieses ermöglicht die dreidimensionale Erfassung des Operationssaales in Echtzeit. Daraus gewonnene Informationen erlauben, im Gegensatz zu den meisten anderen Arbeiten, die Erkennung von Intentionen und Aktionen des chirurgischen Personals, die nicht unmittelbar den Situs betreffen.

Methoden

Das Konzept sowie dessen Implementierung baut auf in der Industrie etablierten Methoden zur Ablaufsteuerung mittels Geschäftsprozessen auf. Hierzu wird die grafische Modellierungssprache YAWL eingesetzt, die eine einfache Definition von Workflows durch Domäneneexperten ermöglicht. Die Abbildung der Prozesse auf ein Assistenzsystem, bestehend aus mehreren Robotern, Eingabegeräten, Kameras, sowie Softwarekomponenten erfolgt auf Basis von Wissen über die Teilaufgaben eines Workflows, sowie über die einsetzbaren Komponenten. Dieses Wissen ist in einer Ontologie modelliert und wird zur Laufzeit durch einen Reasoner klassifiziert. Die Komposition und Konfiguration des Gesamtsystems erfolgt aufgabenspezifisch zur Laufzeit des Workflows. Für jede aktivierte Aufgabe des Workflows verwendet die Implementierung das modellierte Wissen zur Identifikation von Systemkomponenten, die die Ausführung der Aufgaben durch das Personal bestmöglich unterstützen können. Weiterhin ist das System in der Lage die Funktionsfähigkeit aller identifizierter Komponenten zu überprüfen, sowie die Komponenten entsprechend der Aufgabe zu konfigurieren und zu vernetzen. Zur Verfolgung des Arbeitsablaufs lokalisiert das System Komponenten, die in der Lage sind charakteristische Merkmale für die Beendigung einer Teilaufgabe zu erkennen. Die Ausgaben dieser Komponenten werden dann während der Ausführung der Aufgabe überwacht und mit den charakteristischen Merkmalen verglichen. Wird die Beendigung einer Aufgabe erkannt, so wird mit dem Workflow fortgefahren, bis das Ende des Ablaufes erreicht ist.

Als primäre Quelle zur Erkennung der Beendigung einer Aufgabe wurde ein Umgebungsüberwachungssystem entwickelt und verwendet, das auf Basis eines Multikameraverfahrens die 3D Erfassung des Operationssaales in Echtzeit ermöglicht. Dies ermöglicht zusätzlich zu Informationen, die aus dem Situs gewonnen werden können, auch die Einbeziehung des Umfeldes um eine präzisere Erkennung des Arbeitsablaufes zu ermöglichen, sowie das Personal selbst besser in die Ablaufsteuerung mit einzubeziehen und eine intuitive Steuerung des Systems zu realisieren.

Ergebnisse

Im Rahmen der Arbeit wurden vollständige Implementierungen des Kamerakonzeptes auf Basis von Microsoft Kinect™ 360 und One realisiert. Beide Systeme sind in der Lage redundant zu arbeiten. Der Arbeitsraum der Systeme ist ausreichend groß für den Einsatz im Operationssaal. Die Erkennungsqualität von Personen und deren Körperteilen ist von hoher Qualität und im Falle der überlegenen Kinect™ One Implementierung in ca 98% der Zeit mit ausreichender Genauigkeit verfügbar.

Ebenso konnte eine komplette Implementierung der Workflow-basierten Steuerung realisiert werden, die eine ausreichende Ausführungsgeschwindigkeit aufweist. Die Implementierung wurde im Rahmen von Nutzerstudien mit dem NASA Task Load Index und dem User Experience Questionnaire (UEQ) evaluiert. Diese bescheinigen der Implementierung durchweg gute bis hervorragende Eigenschaften im Bezug auf User Experience und Arbeitslast. In allen 9 Fällen konnte das System dem experimentellen Arbeitsablauf erfolgreich folgen. Dieser realisierte die Einrichtung, und Ausführung einer telemanipulierten roboterassistierten Aufgabe an einem Phantom.

Der UEQ Benchmark unterstreicht die sehr guten Ergebnisse und ordnet die Ergebnisse der Nutzerstudie auf den Skalen Attraktivität, Originalität, Durchschaubarkeit und Stimulation unter den 10% der besten Ergebnisse im Benchmark ein. Auf den verbleibenden Skalen Durchschaubarkeit und Steuerbarkeit sind die Ergebnisse 75% der Ergebnisse aus dem Benchmark überlegen.

Diskussion, Fazit und Ausblick

Die in der Arbeit gezeigten Implementierungen von Kamera- und Workflowkonzept konnten ihre prinzipielle Einsatzfähigkeit in einem Versuchsaufbau im Labor nachweisen. Die Ergebnisse bescheinigen den Konzepten vielversprechende Möglichkeiten für die Operationssaalintegration, zukünftige Interaktionskonzepte, Ablaufsteuerung und die Verfolgung des Arbeitsablaufes. Die Implementierung ermöglicht es dem Personal sich außerdem nahezu vollständig auf die Aufgabe zu konzentrieren ohne dem System besondere Aufmerksamkeit schenken zu müssen. Mechanismen zur automatischen Erkennung von Fehlfunktionen und deren Behebung tragen zur Robustheit des gesamten Assistenzsystems bei.

Zukünftige Arbeiten bieten sich im Bereich der automatischen Wissensgenerierung für das System, der weiteren Verbesserung der Kameraalgorithmen und der Übertragung der Konzepte in ein industrielles Umfeld an. Die Arbeit schafft die Grundlage für die Implementierung der Konzepte in einem realen Operationssaal. Zukünftige Arbeiten sollten das Konzept für einzelne oder mehrere medizinische Teildisziplinen implementieren um die Vorteile und Nachteile für die Anwendung zu demonstrieren.

Die wissenschaftlichen Beiträge der Arbeit finden sich in den Bereichen Umfeldüberwachung, Workflow-basierter Ausführung chirurgischer Arbeitsabläufe, modularer fehlertoleranter Assistenzsysteme und der Benutzerinteraktion wieder.

Contents

1. Introduction

Computer assisted systems (CAS) in the scope of medical systems undergo rapid development. Medical robots as a specific category of CAS systems can greatly enhance the medical outcome for the patient and the ergonomics for personnel during interventional procedures such as surgery or radiotherapy [32] [84] [67].

Medical robots exist in the field of surgery, radiotherapy, imaging, rehabilitation and many more. This work especially focuses on surgical robotics. It has been partly created in the scope of *FP7 ACTIVE*, whose application was epileptical surgery, and the national research projects *Single-Port-Technologie für gastroenterologische und viszeralchirurgische endoskopische Interventionen* as well as *Kontextsensitivie Assistenz im aufmerksamen OP*.

Surgical robotics evolved in the mid and late 1980s with industrial robots like the Puma series, that have been adapted to neurosurgical purposes such as brain biopsy [71] or orthopedic purposes such as total knee or hip replacements [102]. It is evident that first attempts, especially in orthopedics followed autonomous and semi-autonomous approaches where preoperative data has been used for planning purposes. Subsequent developments led to minimally invasive robots that heavily relied on the human in the loop, such as the Zeus robot [29]. In the 1990s the da Vinci® surgical system has been developed by Intuitive Surgical, Inc (Sunnyvale, CA, USA) and has been approved by the FDA in 2000. The introduction of the da Vinci® was one of the hugest stimuli for robotics in surgery. The system that was intended for minimally invasive surgery has been used in general surgery, cardiac surgery and various other fields. Parallel to this development, new systems for orthopedic and neurosurgical purposes arose. Newer developments such as the Cyberknife® (Accuray Inc., Sunnyvale, CA, USA) system focus on the enhancement of radiotherapy and medical imaging using robotics technologies. Several recent robotic systems focus on single incision and Natural Orifice Transluminal Endoscopic Surgery (NOTES) methods.

In modern robotics three general control strategies can be identified:

- Autonomous control: The robot acts autonomously, mostly based on preplanned data. This approach is heavily used for imaging robots.

- Semi-autonomous control: The robot assists the surgeon who is in control of the robot e.g. through haptic hints to guide him/her to the target. This approach is commonly utilized for neurosurgical or orthopedic robots.

- Telemanipulation: The robot is directly or indirectly controlled by a human operator. This approach is mainly used in minimally invasive robots.

When a closer look on past and state-of-the-art systems is taken, it can be seen that most if not all existing systems are isolated systems that perform well, but which are also lacking integration and modularisation and often allow for only one of the control strategies. A combination of systems or parts of them, to combine the advantages of the systems or to tighten the integration with the operating room and the medical information system is highly challenging, but may be beneficial in terms of economics, ergonomics and medical outcome.

A trend towards operating room automation and integration is present. For this task there is a need to integrate existing and future sensors, actors and data processing modules. This is challenging due to closed ecosystems of the vendors of digital operating rooms, that mostly integrate only their own devices to allow for a consistent and central control as well as data representation in the operating room. Additionally, the amount of information that becomes available through integration can up to now only be processed to a limited extent. In order to further enhance the capabilities of such operating rooms it can be beneficial to build an operating room that can integrate all kind of devices without being constrained to a single vendor. This is currently scope of several research projects such as [70].

This work contributes to the integration attempts and improves the state of the art especially through workflow and task-dependent system configuration and composition from various system components. The approach is ontologically founded and forms a knowledge-based system that includes process-, system- and component-knowledge allowing for higher autonomy and automatic reasoning in operating rooms. It enables for the semi automatic composition of a configuration of the operating room that suits the needs of the personnel and the target procedure and is based on high-level task descriptions, optical task supervision and a modular framework. The system can follow the surgical procedure based on workflow descriptions and can react to the current situation inside the operating room. The focus is put on surgical robotics but can be generalized to operating room automation.

1.1. Motivation

In the past years, the optimization of the information system structure of hospitals received high attention. In Germany, most institutions are equipped with computer-based information systems that distribute patient information to the whole institution. This is not limited to patient information distribution only, but can also be beneficial to fulfill laws about archiving and to the storage, processing and archiving of radiological information such as medical images. In university hospitals, computer-based information systems can also help to carry out studies through the easy availability of information in a standardized format. The introduction of such systems is usually connected to a high initial investment but pays off due to optimized processes, cheaper archiving costs and better patient care. The integration of hospital information structures usually stops at the operating room. The potential of computer assisted systems is only used to a very limited extend in this scope [35].

It is possible to display information about the patient in the operating room and several assistance systems offer preoperative planning consoles. Some of them can also be connected to the information system of the hospital, which can greatly help the surgeon to perform an intervention. An example for such a digital operating room is the OR1 (Karl Storz GmbH & Co. KG, Tuttlingen, Germany).

However as stated before there is little to no integration of different assistance systems. A lot of potential for patient care, intervention outcome and process outcome is lost due to this lack of integration. With the proceeding automation of the operating room in order to use this potential, the meaningful integration of the information acquired in the operating room and its use for the control of assistance systems is becoming more and more important. Whilst, as stated before, the integration of devices in the operating room is under heavy development by the major medical device vendors, the system composition and data integration into the operating room in order to improve the process itself is still in its early stages [21]. A barrier for this is the complexity of medical interventions that in most cases cannot be completely standardized, which makes modeling and formalization a highly challenging task. Additional complexity is added through multiple available operating methods for a disease, or a treatment, preferences of the operating surgeon and the highly heterogeneous nature of the different devices in the operating room. Several working groups such as DICOM WG24 or IEC/ISO TC/SC 62A JWG9 are following standardization efforts to tackle these issues for future computer assisted systesm.

The current developments result in large sets of available data and information, that could in general serve for the situation aware system composition and configuration. Due to the aforementioned complexity, huge amounts

of the data cannot be adequately taken into account. There is a need for cognition assisted systems that can interpret and integrate the data in order to get the best performance out of the available resources. Additionally, communication structures on which modular assistance systems can be based on are required. It is obvious that a robot or a navigation system can greatly help to improve the overall performance in the operating room when modules of it can be replaced by task-optimized components, or can be reused in several subsystems. Examples for reusability can be using the optical cameras of a navigation system for robot calibration, using the slave robot as a tool holder, or selecting the best fitting kinematics for a specific task. Moreover this can be generalized to data visualization at the right time for the right person in the operating room or to trigger processes in the hospital. In order to allow for such support, the system has to act autonomously in terms of following the workflow of the operation. It has to percept and supervise the situation based on sensor information and processing.

This high-level support system would help the personnel to decrease the cognitive load that is generated by current assistant systems It would allow to improve the procedure through delivering the right information and support at the right time and therefor would help to further improve the processes in the operating room.

1.2. Aim of the work and contributions

This work's aim is to identify factors that can make systems in the operating room context aware, to find suitable methods that can help to achieve this goal and to allow for an optimization of the emerging operating room automation. A strong focus was put on the integration of workflows that are crucial not only in medicine but also in manufacturing. As workflow management systems are strongly integrated into industrial processes, a major aim was to investigate the feasibility of existing workflow management solutions in the sensitive area of operating room control. A control strategy had to be developed that allows to utilize a high-level control strategy for the components used in a system, while retaining the autonomy and internal mechanisms of the used components.

As the focus is put on a workflow-driven system, a crucial point is to follow the workflow. In this work, different camera technologies were used to monitor a predefined workflow without invading the process itself. For this purpose a sensory system that can assist the workflow tracking process had to be developed together with methods that make its usage in the operating room possible and feasible. In contrast to other works that primarily focus on data obtained from the situs itself (e.g. instrument trajectories), a strong focus

was put on the use of environmental data and its importance to the workflow tracking.

There was a need for the development of a framework that helps to implement component-based applications, but that is also able to utilize high-level control strategies, based on the workflow and modelled knowledge. Additionally, a knowledge base had to be established that stores the information about the process, components, systems and compatibilities and that allows for automatic reasoning on this data.

The final aim was to demonstrate the feasibility of the sensor-based approach and the workflow-based controller with a modular surgical robotics system that can execute autonomous, hands-on and telemanipulation tasks and which can be controlled using the workflow system. A prototype that is based on the findings of the proposed approach had to be implemented and evaluated to show the advantages of the approach compared to state of the art systems. The prototype was also used to evaluate how users interact with a context-aware and situation adapted system and how it affects the user experience.

In recapitulation, the aim of the work is to develop and evaluate a control approach that can intelligently compose and configure robotic systems, that is context-aware as well as workflow-driven and therefor helps to enhance current surgical procedures in robot-assisted surgery.

This work's essential contributions are in the fields of multi-sensory optical scene supervision, control strategies for modular systems, operating room automation, cognition and knowledge-based systems in medicine and the modularization of surgical robots.

1.3. Structure of the work

The scope of this work is to deal with workflow-based task execution where the situational knowledge is mostly derived from vision-based technologies. The control strategies are then applied to a robotic system. The described system and methods are divided in three categories:

- Perception system: The primary source of information for the controller. The composition of this system is described together with the methods applied to gain information about geometrical relations in the operating room.
- Workflow-based controller: The central component of the system. It is described how this central component controls and configures involved components based on workflow and system knowledge.

- Application to a surgical robot: A modular surgical robot, developed alongside with this work, is integrated with the workflow-based controller. The system is finally used to apply and evaluate the methods that have been developed for the perception system and the workflow-based controller.

The structure of the work itself is comprised of:

- Introduction: An introduction to the scope of this work is given. The work is motivated and the contributions are outlined.
- State of the Art: An overview about relevant work in this scope is given. The most relevant works are explicitly outlined.
- Background: The necessary background information relevant for the methods used in this work are given.
- Concept and Design: Design decisions taken and the overall concept of the work are depicted.
- Implementation: The methods and implementation for the three parts of this work are shown. Implementation contains a chapter for each of the components (perception system, workflow-based controller and application to surgical robotics).
- Results: The results gained from this work are shown.
- Discussion, Outlook and Conclusions: The Results are discussed, a conclusion about the findings of this work are shown and an outlook on future developments in the scope of this work is given.

2. State of the Art

Medical robots have been established as important tools in surgical centers around the world and the repertory of procedures that benefit from robots is increasing. Numerous approaches are being followed in research and industry to optimize the usability and the behaviour of current systems and to develop new systems to access new applications.

The following chapter gives an overview about other working groups and projects enhancing the usability, system integration and situation awareness of medical systems and especially robots. An overview about current medical robots and new emerging technologies is given together with a review of the current state of the art developing strategies for robotics systems.

The role of the digital operating room (DOR) is emphasized as assistance systems and their integration are important factors for hospitals to improve medical care and to decrease costs. Robots as assistance systems are important parts of such highly engineered operating rooms. Thus the current developments of digital operating rooms are depicted.

Understanding and modelling the process is crucial to provide optimal assistance to the operator in every application. Workflows can serve as models of processes. They already arrived in medicine and it is now possible to detect phases and actions in surgical procedures. In parts, the knowledge of the personnel, especially the knowledge of the surgeon, can be modelled and formalized in a machine-readable way. Several groups are working in the field of surgical workflow and process analysis as well as real-time detection of surgical phases. These works are being reviewed together with the role of workflows in industry, where manufacturing and delivery processes are already automatised through workflow management systems and corresponding process knowledge. It is shown how the problem of formalizing and using the process knowledge acquired is tackled in other works using ontologies to make it usable in workflow management and assistance systems.

At the end of the chapter, an analysis of existing multi-camera 3D techniques and systems is given as 3D supervision of the environment is used as primary and non-invasive source of real time information about the process in this work. Finally, the systems that are most similar to this work are presented and it is described where this work progresses beyond the state of the art.

2.1. Medical Robotics and Robot Control

Medical robots became a part of the clinical routine work. The market leader's Intuitive Surgical da Vinci® system [45] with more than 500.000 procedures per year, more than 3000 installations and more than 8000 peer-reviewed articles regarding the system [58] is a world wide accepted and proven system. In the United states >30% of all hysterectomies [58] are preformed using a da Vinci® robot as shown in Fig. 2.1. The scientific interest in this system is high and studies about new procedures and critical reviews are being released frequently.

Figure 2.1.: The components of the Intuitive surgical da Vinci® S system, source: [123], ©[2014] Intuitive Surgical, Inc

In [24] the authors present a retrospective single-surgeon study and compare robot-assisted to laparoscopic nephrectomies. The authors observed comparable treatment results but longer operating times and significantly higher costs when using the da Vinci® robot. They conclude that for this specific procedure, the da Vinci® robot does not offer benefits. However, they highlight the better ergonomics, the high instrument dexterity and the short learning curve when using a da Vinci® robot. Higher operating times are explained with port planning and installation times of the robot and a lack of medical tools needed for the specific procedure. The authors conclude with a need for a wider range of robotic tools and raise a discussion about the monopole of Intuitive Surgical what could possibly keep the costs high.

In contrast to this, the authors of [101] review the da Vinci® robot in radical prostatectomy, one of its main application domains and come to the conclusion that robot-assisted radical prostatectomy in comparison to conventional procedures shows shorter OR times, shorter stays in hospitals, decreased blood loss and promising results in terms of medical outcome. In [133], the authors carried out a case control study with 54 patients that underwent a

radical cystectomy. In the authors' study, the blood loss rate and length of hospital stay were decreased in median compared to the control group. They also discovered that in median, the robot-assisted operations had a longer duration compared to the control group.

Apart from urology, another popular field where the da Vinci® robot is being used, is cardiothoracic surgery. The authors of [96] reported about their prospective study with 112 patients that underwent a mitral valve repair using the da Vinci® robot. Several surgical teams were involved in this study. Again they showed higher procedure times but decreased need of blood transfusion as well as lower complication and mortality rates. Like in laparoscopic heart surgery there was no need for a sternotomy. In [132] 225 laparoscopic aortic procedures have been performed using the da Vinci® robot. In the discussion, the authors state that robotics helped to decrease the clamp time which is one of the biggest disadvantages in laparoscopic aortic procedures. Also they stated advantages for obese patients and shorter stays in hospitals. They also state that there are patient groups that cannot undergo both robotic- as well as non-robotic laparoscopic surgery. Whilst this overview about some application fields of the da Vinci® as the most popular robot to date cannot give a complete review about advantages and disadvantages of this system it shows how actively available robotics technology is applied to different fields in medicine with varying results and how important the flexibility of robots in medicine is. There is no general answer to the question whether robots should be used for medical procedures or not. There are hints that da Vinci® assisted operations are superior to conventional procedures, e.g. when regarding blood loss, but there are also fields there there are little or no advantages through the use of the da Vinci® robot.

The following paragraphs gives an overview about the experiences of medical and technical users with medical robotics in general. It is shown how the experience with the da Vinci® robot in hospitals is like and how other groups try to enhance robotics technology in medicine. Other application fields like orthopedic surgery, neurosurgery or single port surgery are depicted together with researchers' views about medical robotics.

In 2002 the authors of [138] carried out a small scale study with synthetic tasks to evaluate the learning curves of robotic and non-robotic laparoscopic surgery. They demonstrated that not so experienced surgeons can learn faster using a robot.

Neumuth et. al in [92] evaluated, how the da Vinci® telemanipulator affects the surgical workflow. The role of medical workflows is shown and discussed in subsequent paragraphs of this state of the art analysis. The authors of [92] used Surgical Process Models (SPM) and introduced Resource Impact Profiles profiles to analyze the Surgical Process Models. They applied their method

to 24 nissen fundoplications which is a procedure to surgically treat gastroesophageal reflux disease. 12 of the 24 procedures were carried out with the da Vinci® telemanipulator. They did not find significant impacts of the da Vinci® system to the surgical process, but they found differences in transition probabilities between tasks, performance repetitions and performance durations. They conclude with a limited impact of the da Vinci® system on the workflow when used for performing nissen fundoplications. However they expect a superiority of the system when using it for more complex procedures and they showed that surgical process modelling is applicable to assess the impact of a surgical assistance system.

The articles of Taylor [124] and Beasley [18] review the situation of medical robotics in general. They give an overview about past, current and the authors' perspective on future robotics systems. Taylor outlines the ability of a robot as a computer-integrated system to

> significantly improve surgeons' technical capability, either by making existing procedures more accurate, faster, or less invasive or by making it possible to perform otherwise infeasible interventions (Taylor, 2004, source: [124], p.2, ©2004 IEEE)

He outlines that a computer-integrated system can support the surgeon through additional information that can be presented in an appropriate way (e.g. virtual fixtures). Additionally a robot can collect information about the procedure itself to enhance the safety. He outlines the importance of surgical planning that can both support follow up treatment of the patient and the surgical procedure as well. He emphasizes the importance of modular systems that can be put together to form a therapeutic system and the importance of modelling tasks, patients and patient groups, as well as the environment. He also talks about hands-on robots that can be used in a cooperative way and that have been observed to be very natural in use by the surgeons.

Beasley gives an overview about medical robotics that is not specific to a single medical domain. He emphasizes neurosurgery, orthopedic surgery, general laparoscopy, percutaneous access, steerable catheters, radiosurgery, emergency response, prosthetics and exoskeletons, and assistive and rehabilitation systems by presenting several systems in all of these domains. He shows how many commercial systems are available up to date. Beasley also does not neglect research systems and talks about capsule robots, the Raven™ II, Mirosurge, Neuroarm and several other systems. Some of them are described in the following. Beasley concludes with

> In surgical robotics, there has been a trend away from autonomous or even semi-autonomous motions, and toward synergistic manipulation and virtual fixtures (Beasley, 2012, source: [18], p.10, ©2012 Ryan A. Beasley)

In [47] and in [68], the medical robot - Mirosurge - of the DLR, shown in Fig. 2.2 is described. This system consists of three lightweight robots that can be mounted to the operating bed. Two of the robots can hold the surgical tools and the remaining robot is used for positioning of a 3D endoscope used in bimanual telemanipulation. The concept is similarly used for the robotic system used in this work. It enables for a flexible placement and a modular design of the assistance system. The Mirosurge system as a sophisticated telemanipulator provides haptic feedback and custom instruments similar to the Endowrist instruments of the da Vinci® robot. Additionally, the Mirosurge robot, whose primary application field is minimally invasive surgery, can be used as a soft robotic system and the authors investigated the robotic system use in other domains such as robotic laser osteotomy.

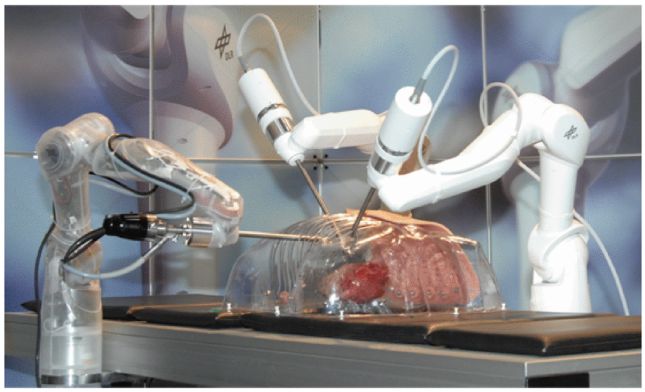


Figure 2.2.: The slave side of the DLR MiroSurge system, source: [68], p.1589, ©[2009] IEEE

A different approach is followed by with the Raven-II™ robot described in [50] and depicted in Fig. 2.3 by which the authors offer a completely open research platform for minimally invasive surgery. The authors opened both hardware and software and use ROS [104] on top of Linux as an open base for developments.

In [86], the author presents a multi-purpose robotic system platform that can be controlled through simple workflows and includes a basic environment supervision system. The robots used in this system are KUKA LWR IV robots [22] which are related to the design of the manipulators of the DLR Mirosurge system. This system can be used in soft robotics, telemanipulation, and autonomous modes. Being the predecessor of the system developed during this work, [86] is one of the most relevant works for this thesis.

Other systems in research are following the single port surgery approach, which allows for a single incision only approach and has the potential to further decrease invasiveness in minimally invasive surgery. To even further reduce invasiveness, endoluminal or transluminal approaches (NOTES) are

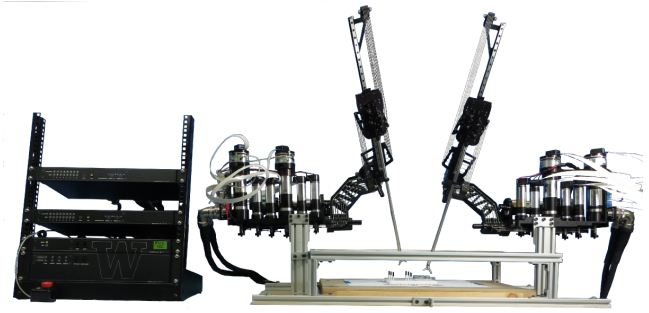


Figure 2.3.: The slave side of the Raven-II™ system source: [39], ©[2013] Applied Dexterity

followed, where natural orifices are used to access the body. An example for such a system is the STRAS system which is depicted in [38]. This system is a flexible robot which can be introduced through a natural orifice and that can manipulate two instruments and an endoscopic camera. In neurosurgery a trend towards intra-operative imaging is present as shown in [74], where the authors present a new fully MRI compatible robot for neurosurgical interventions.

In conclusion, it can be said that medical robotics arrived in various medical disciplines. There is evidence that some procedure can benefit from medical robotics while other procedures do not. Up to now, there is little to no tight integration of medical robots with information systems or the operating room itself, making the robots solitary but high performance assistance systems for the surgical personnel.

2.2. Digital Operating Room

As an assistive system a surgical robot can be put in the same category as imaging systems and navigation systems. When planning and installing such systems, a modern infrastructure is crucial for the improvement of ergonomics, patient care and economics of a hospital. Several work groups are investigating how high tech assistive systems can be integrated into an operating room. Technologies called Hybrid-OR combine an operating room with one or more imaging systems. These paragraphs discuss the need for future operating rooms, experts view and research carried out.

The authors of [69] line out data acquisition and interpretation, taking into account the workflow, environment supervision and introduction of autonomy into the operating room. They state that these enhancements can improve intuitiveness and decrease the surgical workload, probability of failure and

times to perform actions. The authors explicitly identified robotics as a part of the future OR and emphasized that future research will and should be directed towards autonomy for certain tasks like suturing. Emphasise is given to context sensitive interfaces, adaptable operating room settings, specific to the personnel and the procedure, the better integration and improvement of imaging and visualization technologies as well as the integration of the workflow and patient data. They conclude with the need for sophisticated assistance systems as well as comprehensive data collection and interpretation strategies This is well aligned with [77], in which the authors identified sophisticated user interfaces and modelling as major tools for improving current operating rooms. For the digital operating room (DOR) they defined an evolutionary growth path, which is depicted in Fig.2.4 until 2025, which includes the introduction of modelling, imaging, simulation, intelligent systems, workflow management and visualization technologies alongside others. They outline the importance of process and patient specific modelling as well as a tight integration of the systems. Also in [21] the importance of process

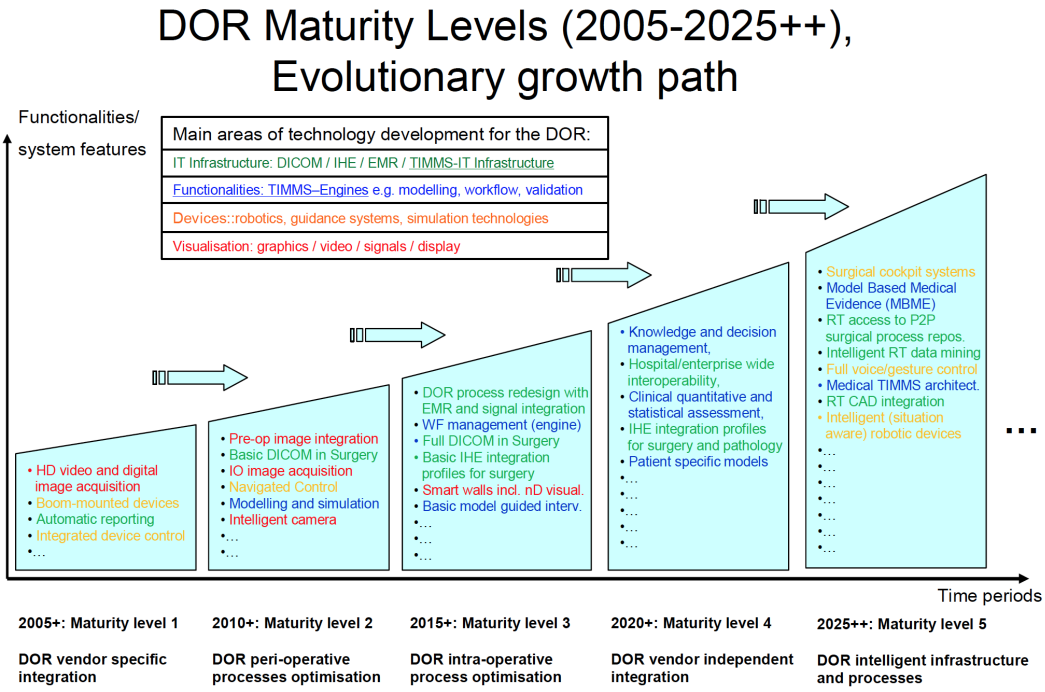


Figure 2.4.: The maturity levels of the Digital operating room as stated by Lemke et al., source: [77], p.4, ©[2013] SPIE

models, ergonomics, workflows and robotics for the development of future operating rooms and devices is shown. In [126] the authors introduced the OpenIGTLink protocol for the integration of systems for image-guided therapy, which is well accepted in the scientific community showing the need for

a common communication protocol that can transmit images, transformations and data in general.

In 2004, a workshop about the operating room in 2020, called OR2020, has been held. This workshop received international attention. The workshop was intended to identify the requirements for future operating rooms and a report was released in 2005 [35]. The report concludes with the need for standards, plug-and-play architectures, integration of the workflow, improved patient safety, improved imaging and the need for infrastructures enabling telecollaboration.

In conclusion, one can say that most of the authors agree that medical robotics together with process modelling and workflow integration as well as sensor equipped operating rooms are highly important factors for the implementation of the operating room of the future.

2.3. Workflows in medicine

As shown in the previous section, the medical workflow is of big importance for the optimisation of assistance systems in terms of ease of use, economics and performance. This applies to industrial systems as well. The following section shows how workflows are identified, modeled and used in medicine, research and industry.

In [128], a survey about Business Process Models (BPM), a common way to represent workflows in industry, is presented. BPMs, in contrast to workflow management (WFM), also support during the analysis phase and help in improving the process itself. The authors identified different notations for BPMs such as Business process model and notation (BPMN), Petri-Nets, or the Universal Markup Language (UML). They show the typical BPM lifecycle including design, implementation and adjustment, which is depicted in Fig. 2.5. It is shown that BPMs can be used for process mining to improve a process and to build process-aware systems. Enterprise resource planing systems (ERP), which are widely used in industry, also belong to this category and can be used together with BPM. Three categories for process modelling languages are identified:

- Formal languages such as Petri Nets that can be easily used for analysis tasks.
- Conceptual languages such as BPMN or UML that are well-suited to precisely model every part of the process but usually cannot be directly executed.

- Execution languages suchs as the Business process execution languages (BPEL) that are usually proprietary languages of vendors offering BPM systems.

In this work the open source YAWL language, which is described in [10] and [129], has been chosen as workflow management tool, due to its flexibility, completeness and ease of use. The authors built a complete open source workflow management system that is based on the Apache Tomcat webserver. YAWL is not intended to be a competitor of commercial systems but can easily be integrated into own developments, allows for a flexible adaption to the developer's needs and supports most of the available workflow patterns [129].

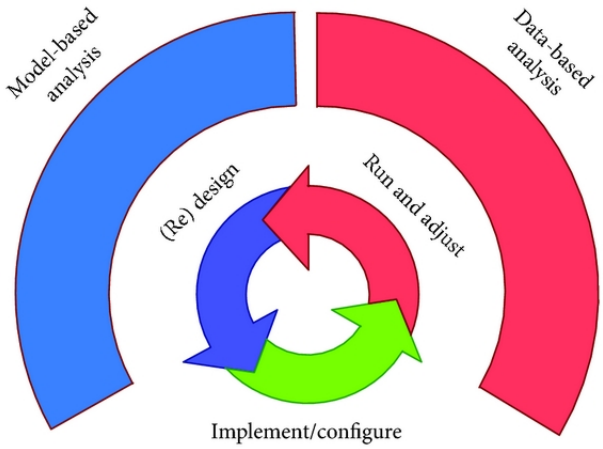


Figure 2.5.: The typical lifecycle for Business Process Models, source: [128], p.5, ©[2013] Wil M. P. van der Aalst

The authors of [25] introduce SMACH for ROS that allows for task execution in complex robotics systems. However SMACH is not a workflow management system and is limited to finite state machines only. Each SMACH configuration requires additional programming of the developer. SMACH intends to enable rapid application development with ROS. In [37] several workflow management systems are being compared to each other with repsect to scientific workflow management. The authors conclude that it is unlikely to achieve a standardization like BPEL in business process modelling due to the variety and different needs in different research domains.

In medicine, several attempts have been made to integrate workflow management to information systems such as [42], where the authors developed and implemented a communication structure to connect a system that manages generalized SPMs with an operating room bus in order to enable adaptive behavior of medical devices. They demonstrated the feasibility of this approach. However devices appear to be used in a standalone mode where

relations between the devices are not given priority to. Another approach to improve medical systems through workflows is shown in [141], where the authors propose a medical information system that is driven by the workflow management system YAWL. They state that their approach improved the process and helped to monitor it. They also discovered easier integration and maintenance as well as a faster development.

To enable workflow-driven distributed systems in medicine, the authors of [80] facilitated generalized SPMs and translated them into the YAWL language. An integration into a distributed system itself is not shown in this work. A thorough investigation of the clinically relevant surgical process models is given in [73]. The authors state that model-based systems are an important technology to enable next generation computer assisted surgery (CAS) systems. They relate to [93] where a SPM is defined as:

> Surgical process models are models of surgical procedures that reflect a predefined subset of interest of the real intervention in a formal or semi-formal representation (Neumuth, 2007, source: [93], p.1, ©2007 CARS)

After a literature review of 46 articles, they came to the conclusion that an ontology is a crucial tool to formalize collected information about the process in order to have a common language that can be processed. They identified the possibilities of observer- and sensor-based data collection, whereas in a, observer-based approach. a human annotates clinical data and in a sensor based approach automatically acquired data like videos can be analysed. They identified that SPM are mostly applied to minimally invasive- and neurosurgery. The authors state that in the future, SPMs have to be integrated with multiple sensors to allow for the detection of e.g. tasks within a procedure.

In order to allow for a better integration of SPMs, which can be hard to process, into information and assistance systems, the authors of [91] propose a four level approach to formalize SPMs. They demonstrated the feasibility of their system using four example domains like nissen fundoplications. In [23], the authors show how the log files from executed workflows can be used to train a Hidden Markov Model (HMM) with which they were able to track the state of the procedure. This approach demonstrates, how knowledge can be extracted from a set of observations using workflows. Moreover, this approach shows how a system following a workflow can be improved through data mining in workflows.

It is obvious that workflow recovery is of great importance for context-aware systems in order to keep track of the performed workflow. In [12], the authors

present their novel technique for workflow recovery using an averaged workflow, showing the feasibility of workflow recovery in minimally invasive surgery.

Regarding surgical workflow analysis and detection, various works in several domains can be found. In [121], the authors analysed the workflow of functional endoscopic sinus surgery and formalized it using ontological methods. In [98], the authors modeled the laparoscopic cholecystectomy workflow using observations from 16 procedures. The authors used HMMs and Dynamic Time Warping (DTW) to build a workflow, which could then be used for online workflow detection. An approach for the same procedure using data from a navigation system only is shown in [122]. In [134], [26], [119] and [55], workflow detection methods are shown in sigma resection, laparoscopic cholecystectomy and tracked needle interventions. The authors of [79] investigated the influence of missing data to workflow detection. They trained generalized SPMs (gSPM) from individual SPMs (iSPM) of 100 cataract surgeries and simulated the loss of sensor data. They analyzed how the number of iSPMs used for the training of the gSPM influences the detection quality in the process after a data loss and they showed that finding the actual chosen path after a data loss is possible.

The literature shows that Bayesian methods, Dynamic Time Warping and Hidden Markov Models are predominant in surgical workflow detection. Recent works use Random Decision Forests such as in [119]. It is also striking that most of the work merely focuses on the intervention itself using information about used tools and tool position only, while neglecting the environment. An exception is shown in [100], where the authors use a 3D camera system to supervise the operating room. A multi 3D camera system with 3D reconstruction capabilities has been used and HMMs have been trained to follow the steps of the procedure. The authors demonstrated the feasibility of this approach for high level tasks such as patient preparation, or cleaning. Detectable situations and the corresponding representation from the system's point of view are shown in Fig.2.6.

In [88] and in the dissertations of J. Münchenberg [89] and O. Schorr [111], the authors present the KasOP system, which is a planning system for robot-assisted surgery. The authors use modelling and workflow techniques to build a plan for execution with a robot and evaluated the system in maxillofacial surgery. The system allows for integration of various hardware and software modules through middleware technologies. The planning procedure is independent from the intraoperatively used systems. However, the system needs knowledge of the devices that can be used and plans all required device actions prior to the execution of a plan. In one of the last steps of the planning process, an execution plan for the devices that can be

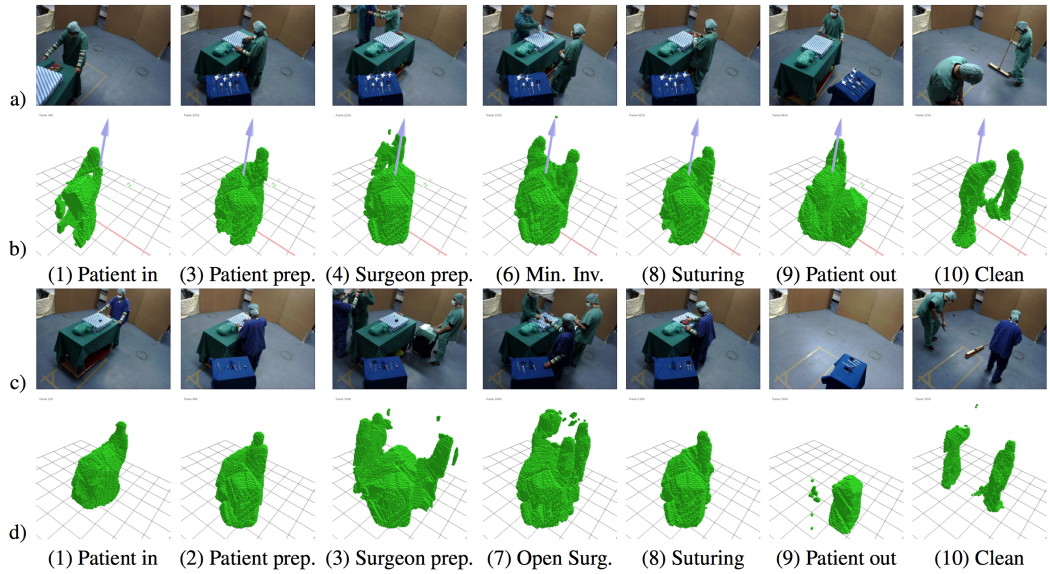


Figure 2.6.: Detectable situations and system output of the system by Padoy et al., source: [100], p.3, ©[2009] IEEE

used is computed. Additionally the system can be used intraoperatively as an assistance system for the surgical personnel.

In conclusion, the literature gives strong indications for the power of surgical workflow analysis and detection using generalized surgical process models. It is already shown that both visual data from the operating room and data collected from the operation itself can contribute to the analysis and detection of a workflow. Systems that integrate process modelling and medical workflows into the operating room, into surgical planing and even into robotics surgery, have been demonstrated.

2.4. Formalization of knowledge and reasoning using ontologies

As seen before, the formalization of process models and workflows is crucial for automatic processing and execution. Formalisation can be cumbersome. A classical way is to store the knowledge in relational databases. This approach requires an almost complete understanding of the process prior to the modelling and can be inflexible. A more user-centered approach is the modelling of knowledge inside ontological systems. When formal logic like OWL-DL [1] is used, this approach allows for automatic reasoning to identify new or hidden knowledge and to allow for decision making. Such methods fall in the category of semantic modelling. The literature shows several approaches

to formalize process knowledge or to allow for decision support in robotics and medicine using semantic approaches.

In [131], the authors show how workflow execution can be ontologically supported in case of missing information. The authors show a decision making support process to help with the selection of sub workflows and demonstrate the feasibility of their approach in beach cleaning after an oil accident on the sea. One of the latest works in the field of medical ontologies is [15], where the authors designed an ontology to store information about different human body systems. They use this system together with Bayesian networks to support information exchange during medical diagnosis.

Also specific interventions or symptomatic ontologies have been designed like shown in [63], where the authors designed an ontology containing knowledge about epileptic seizures to support automatic classification of these pathophysiologies. In [64], the authors modelled process knowledge in the field of laparoscopic surgeries and used the ontology for intraoperative situation interpretation in order to enhance the capabilities of assistance systems.

In [107], a system for semiautomatic knowledge extraction from recorded clinical processes is shown. The authors show the ability of the system to help for the identification of the occurrence of errors in lung cancer treatment.

The following two approaches show semantic systems for robotic purposes. KNOWROB [125] is a system that enables knowledge processing in autonomous mobile robotics and allows for the usage of environmental sensors in order to let the robot perform actions. In [16], an approach that allows for self configuration in social robotics using ontologies is shown. The authors applied their approach to social robots in healthcare. Both approaches have in common that they use the ROS framework to control the used robots. In [48], the authors present a survey about the usage of ontologies in surgical robotics. Relevant ontologies were in the field of rehabilitation robotics, neurosurgery and orthopedics, but no generic process modelling system for surgical robotics has been identified. The authors state that

> Ontologies can help to build entire platforms for new robotic services (Haidegger, 2013, source: [48], p.5, ©2013 IEEE)

In conclusion it can be said that ontologies allow for a formalization of knowledge in various fields. Recent work shows the feasibility and usability of semantic systems to support processes in medicine and (medical) robotics.

2.5. 3D Camera Systems

3D cameras have received high attention in science since the release of the Microsoft (Redmond, WA, USA) Kinect™ 360. The high interest in the Kinect™ 360 served as an accelerator for 3D segmentation techniques in pointclouds, which made the Point Cloud Library (PCL) a mature library for 3D point cloud processing [108]. 3D cameras allow for the scene supervision with spatial data, when only one camera is used. This simplifies calibration and postprocessing of the data. Systems have been around before the release of the Microsoft Kinect™ 360, such as Time-of-flight cameras (ToF), or stereo cameras. This paragraph gives an overview about the application of 3D camera systems for supervision and medical tasks.

A Tof, a Kinect™ 360, a a Kinect™ One and an ART tracking system camera can be seen in Fig. 2.7

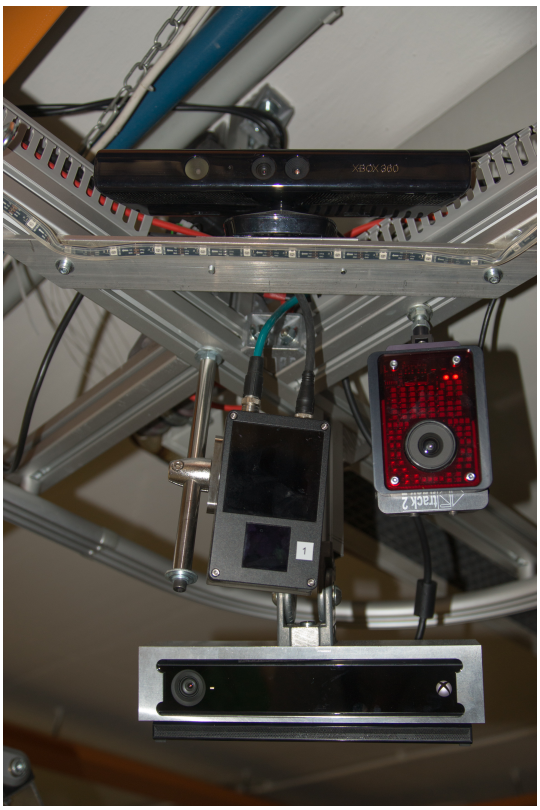


Figure 2.7.: Top: Microsoft Kinect™ 360, Mid right: Advanced Realtime Tracking GmbH ARTtrack2 (Weilheim, Germany), Mid left: PMDtechnologies s3 (Siegen, Germany), Bottom: Microsoft Kinect™ One

As the Kinect™ 360 offers good quality 3D point cloud generation for a

reasonable price, due to its consumer-oriented design, various scientific works thast use the camera can be found. One of the most regarded works is presented in [59], where the authors present a system in which the Kinect™ 360 is moved by a human operator allowing for 3D reconstruction of the scene in real-time. This process utilizes the processing capabilities of modern graphics adapters and allows for a 3D representation of the environment with a single camera that can e.g. be mounted on a mobile robot. An example output of the system can be seen in Fig. 2.8.

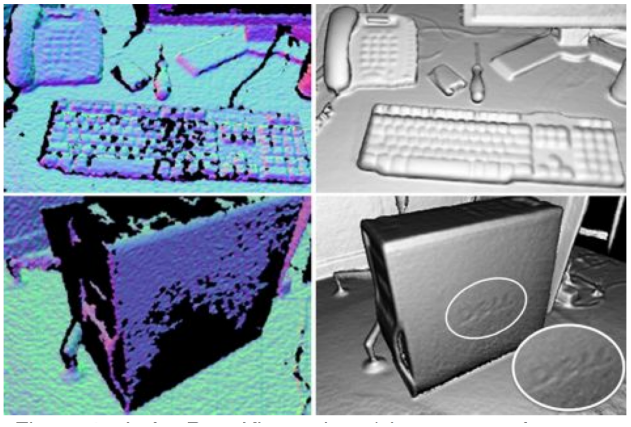


Figure 2.8.: Left: The Raw data of the Microsoft Kinect™ 360. Right: The reconstruction of the Algorithm of Izadi et al., source: [59], p.3, ©[2011] ACM, New York, NY, USA

Other works combine multiple cameras that are rigidly calibrated with respect to each other to perform real-time monitoring of a large workspace using the Kinect™ 360 camera. But also systems utilizing multiple 2D cameras or ToF cameras can be found. In [34], a ceiling-mounted camera system for the supervision of a swinging service robot is presented. The system is using 2D cameras that are registered with respect to each other. The authors were able to extract volumetric data from the supervised scene but did not compute a complete scene representation in real-time. In [139], the authors depict a system that fuses stereo 3D vision with ToF-based 3D vision. They state that stereo vision is still challenging due to texture dependence and proneness to occlusions. ToF cameras on the other side do not offer the resolution of stereo camera systems and suffer from random noise. The authors achieved a 3D representation of the scene that combined the advantages of both systems. Their approach is able to build complete 3D models that allow for the compensation of noise introduced by ToF cameras. A newer work using the Microsoft Kinect™ 360 cameras in a multi camera configuration is shown in [40]. The Kinect™ 360 as an active structured light system suffers from noise introduced through the use of multiple Microsoft Kinect™ 360 cameras that project their pattern on the same scene. The work quantifies how much

noise is introduced by the use of several cameras. The authors use a Kalman filtering approach to track objects in the 3D space. Based on the position of the object, the 3D camera with the highest probability to properly track the object is activated in order to avoid the noise introduced through the use of multiple Microsoft Kinect™ cameras. In [90], a system for 3D reconstruction using multiple Kinect™ 360 cameras is shown. As multiple Kinect™ 360 cannot be synchronized to each other in hardware the authors proposed an approach based on temporal interpolation. They demonstrated the feasibility of their approach with a single person moving within a rectangular experimentation area, which can be seen in Fig. 2.9. The scene itself was uncluttered.

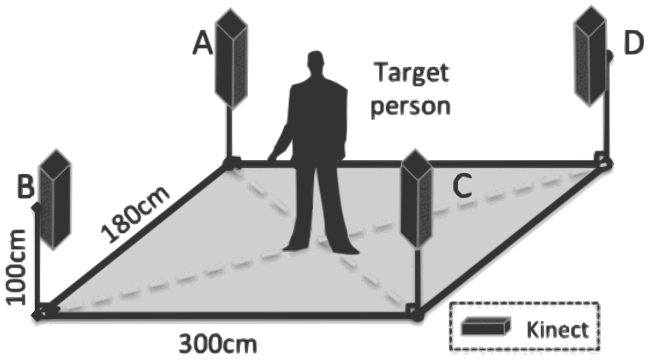


Figure 2.9.: The experimental setting of Nakazawa et al., source: [90], p.472, ©[2012] IEEE

An approach to use depth cameras for user interfaces is shown by Microsoft research in [135]. The authors show a room that is equipped with multiple projectors and depth cameras and use the system for Natural User Interaction (NUI). They detect interactions with the projected environment and allow for the interaction with the room. The Kinect™ 360 itself is designed as input device for games played on the Microsoft Xbox™ 360. It allows for interaction with the game using the own human body as a controller.

Not only because of this, there is a need for accurate body tracking algorithms. Several approaches can be found in literature. OpenNI, which was developed by Primesense (Tel-Aviv, Israel) was an open source framework for the development of user interfaces with the cameras featuring the Primesense depth sensor, such as the Kinect™ 360 camera. NITE (Natural Interaction Technology for End-user) was a framework to allow for full body tracking with OpenNI. Primesense was bought by Apple (Cupertino, CA, USA) in 2013 and OpenNI and NITE are no longer available. The most popular algorithm for full body tracking is integrated into the official Microsoft Kinect™ SDK. This algorithm is described in [114] and is based on a huge database of human postures and human statures. The approach is based on the Random Decision Forest classifier, which is highly regarded as a high quality classifier.

Whilst this approach provides good results for full body tracking, as can be seen in Fig. 2.10, and can be freely used in own projects, it is not available as open source software and can only be used on Windows-based systems. This leaves space for open source Random Decision Forest based classifiers like described in [30] and built in to PCL.

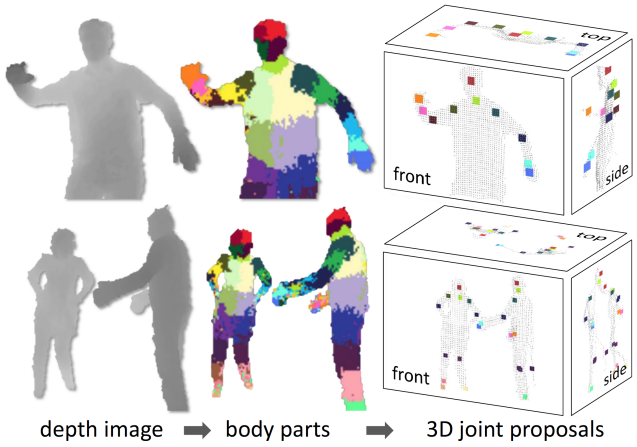


Figure 2.10.: The body part recognition from Shotton et al. From left to right: Depth image, colored body parts, joint proposals, source: [114], p.1, ©[2011] ACM, New York, NY, USA

A comprehensive overview on human tracking and pose estimation reviews prior to the availability of the Kinect™ 360 is given in [53]. The authors show that also before the release of consumer 3D hardware several scientific groups were working on single camera human pose estimation. The authors state that marker-based tracking systems are invasive and show that most of the reviewed approaches are based on particle filtering. It is shown that 3D posture tracking is also possible from 2D views.

In [142], a skeleton tracking approach, based on the data of multiple Microsoft Kinect™ 360 is shown. The authors apply a particle filter approach to compute the posture of a human from the 3D data acquired. Using multiple Kinect™ 360 camers, the authors were able to perform continuous tracking of a walking human even though 360 degree movement was performed, which is a domain where OpenNI and the official Microsoft approach fail.

A persistent problem for multiple view and 3D techniques is camera calibration. 2D as well as 3D cameras usually require intrinsic calibration in order to estimate the position of the device. For a Kinect™ camera, multiple calibration problems can be identified:

- Intrinsic calibration of the RGB camera
- Intrinsic calibration of the IR camera

- Depth calibration
- Extrinsic calibration of RGB to the IR camera

If multiple cameras are used also an extrinsic calibration between the cameras is necessary.

In [105], Raposo et al. show an approach, based on the popular work of Herrera [54], to perform all four necessary calibrations for a Kinect™ 360 camera. The authors used a checkerboard-based approach and were able to improve the calibration of the Kinect™360 camera using only 6-10 images. They were able to reduce the translational error to <10 mm and the rotational error to <1 degree. For extrinsic calibration of multiple Kinect™ 360 cameras, multiple works can be found. Two exemplary approaches are shown in the following. In [66], an approach is presented that allows for a calibration of multiple Kinect™ 360 cameras and 2D cameras to each other. The authors show an approach where 3D cameras can be combined with 2D color cameras. In addition, their approach allows for a depth correction of Kinect™ 360 cameras. The work presented in [81] utilizes a set of Kinect™ 360 sensors for vehicle scanning. The authors present an approach to calibrate the Kinect™ 360 cameras using a checkerboard for extrinsic and intrinsic calibration. The extrinsic calibration is performed using the planar estimation of the checkerboard from multiple views using the infrared image of the Kinect™ 360 only. Another work that tackles the problem to register multiple Kinect™ 360 is shown in [136]. The authors use a checkerboard camera calibration approach that is applied to the infrared cameras (IR). After an initial alignment, they use an iterative closest point (ICP) [20] approach to refine the extrinsic calibration.

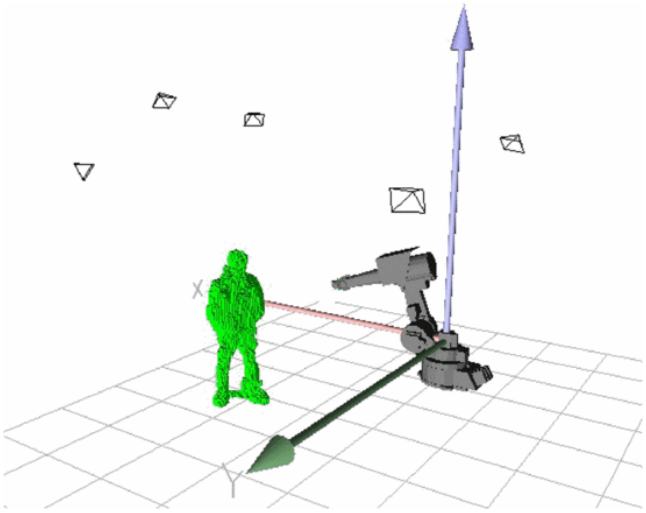


Figure 2.11.: The Voxel-based human detection from Stengel et al. with respect to an industrial robot, source: [120], p.1389, ©[2012] IEEE

Applications for such 3D supervision methods can be found in mobile robotics, industrial robotics, gaming, human machine interfaces (HMI) and also in medical applications. Some applications are shown in the following. In [120], the authors present a system consisting of multiple RGB cameras to perform safe human robot interaction using an industrial robot. The system is able to detect humans using a voxel-based approach. The voxel-based representation with respect to an industrial robot can be seen in Fig. 2.11. Robots can be found within the scene using their CAD model. An application of a multiple Kinect™ 360 approach in rehabilitation, where the Kinect™ 360 is heavily used due to its ability to track motions without markers, is shown in [78]. The authors propose a system consisting of several Microsoft Kinect™ 360 cameras that can be installed at a patients home without the need of a checkerboard for calibration. The authors use a Kalman filter to fuse the skeleton tracking data of multiple camera in order to acquire a precise full body model of the patient including the skeletal configuration. However it seems that in the presented stage, the system cannot cope with multiple people in the field of view of the cameras. Another multi Kinect™ 360 approach is shown in [13], where the authors built a CUDA (NVIDIA, Santa Clara, CA, USA) powered approach for real-time mesh reconstruction of moving objects in the field of Tele-Immersion. In [31], a multi Kinect™ approach for gesture and postural interaction is presented. The system is able to track multiple users and is also able to fuse the tracked skeletal configurations in multiple views. The presented system purely relies on the postural data and therefor requires full body tracking to be available. The target environment in a living room can be seen in Fig. 2.12.

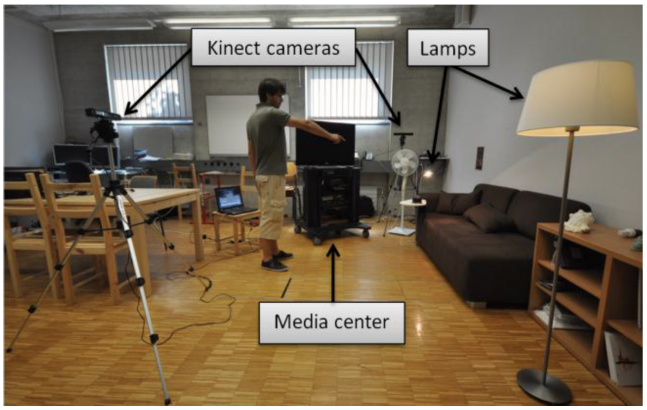


Figure 2.12.: The Multi Kinect™ 360 approach of Caon et al. in a living room, source: [31], p.11, ©[2011] IARIA

A work in the medical context is presented in [72], where the authors utilize an array of 16 2D cameras to reconstruct the 3D environment of an intervention room in real-time. The purpose of their work was to avoid collisions with a robotized C-Arm that was installed for interventional radiology purposes,

and allow for a higher safety and faster movements of actuated devices.

The field of 3D capturing techniques was heavily catalysed through the Microsoft Kinect™ 360 and its successful hack by the open source community. Work prior to the release of the Kinect™ 360 usually either had to cope with the incompleteness and computational complexity of 3D stereo techniques and other reconstruction methods from a set of synchronized 2D images, or with the noisy and low resolution images of ToF cameras. The Kinect™ 360 provides high resolution depth images that allows to build complete 3D scenes and to even accurately infer human postures in real-time. This work was adapted to various applications in user interaction, rehabilitation, industry and medicine. With the release of the new Kinect™ One based on ToF technology that claims to offer a more accurate depth image, it can be expected that existing applications can be improved. The power of 3D scene supervision and body tracking has been successfully demonstrated in various works.

2.6. Most similar works and progress beyond the state of the art

This work presents a complete system for a workflow-guided intervention system that is heavily supported by optical supervision. However, several topics have to be addressed. Similar works can be found in the fields of robotics systems, semantic processing and 3D supervision as well as tracking technologies.

In terms of the target robotic system for the developed methods, the most similar work is the DLR Mirosurge robot described in [47] and [68]. The system is similar in arm configuration and it is intended to be integrated in interventional systems for minimally invasive surgery or orthopedics. The system design as can be judged from the literature is not intended to allow for flexible usage. The kinematics however is excellent in terms of flexibility as a multi-purpose robotic manipulator for surgical purposes.

The work of Taylor [124] shows how crucial modular robotics in medicine is and how important situation awareness can be, but no system is shown being able to fulfill the needs described in the article. In [100] Padoy et al. show how the supervision of the intervention room can help for context-aware systems. The system itself is based on point cloud-based processing and focuses on the classification of operating room configurations. 3D supervision techniques are used in [72] by the same workgroup for collision avoidance. Despite showing decent results, both systems are not included into a modular and flexible system for interventional assistance.

A system very similar to the 3D supervision system developed in the scope of this work can be found in [31], where a system is shown that allows for multi user tracking based on skeletal information which does however not appear to be able to work with a point cloud-based approach only.

Operation planning and execution based on workflows to allow for cognitive and context-aware systems in the operating room is shown by Münchenberg [89] in maxillofacial surgery. The author's approach allows for an operation planning on a high level, down to the execution on real hard- and software within the operating room. The thesis of [86] is most relevant to this work, as a first medical telemanipulation system that is driven by workflows and guided by optical techniques is shown. The system does allow for the execution of predefined workflows with actions being defined prior to the execution on well known hardware. The system uses very basic techniques for operating room supervision.

This work progresses beyond the state of the art in three categories:

- **Medical robotics:** A new dynamically reconfigurable approach for modular robotics is presented and applied to a medical robotic system that allows for a semantic high level controller but leaves the complete execution to the robotics system.
- **3D Supervision:** A multi camera approach using Microsoft Kinect™ 360/One cameras is shown that allows for a point cloud-based extraction of users from the whole scene but also allows for skeleton tracking and fusion of the detected users. The proposed approach is independent from the amount of cameras used and can in general also be applied to other 3D Camera systems. It allows for a point cloud-based execution when no tracking information is available and can complete the user model as soon as tracking information is available. Additionally, a new approach for the registration of multiple RGB-D cameras is shown that relies on both the 2D and the 3D information acquired from the scene
- **Cognitive workflow execution:** A novel approach to execute high-level workflows in the medical context is shown that allows for automatically device identification and that can compose the system from available components in real-time. This allows for a reconfiguration of the system at run-time and the detection of faulty subsystems. The approach, based on existing workflow technologies in general facilitates workflow mining technologies.

3. Background

This chapter shows the underlying principles used in this work. Specially emphasized are the principles of 3D camera technologies, the basics of workflow management, modular robotic systems and knowledge management.

3.1. Modular Robotics

Complex systems such as robotics systems underwent a paradigm change in the last years [85]. There is an evident trend from *big binary* towards modularization of systems. This facilitates the reuse of system parts as well as easy maintenance and introduces the possibility to distribute the system among homogeneous or heterogeneous computer platforms, to allow to combine the computational powers of the computers or to utilize the advantages of all used platforms. This approach also has drawbacks, such as the need for underlying frameworks that allow for modularization and the frameworks ramifications such as constraints in programming, varying realtime capabilities and the increased complexity of such systems.

If every module lives in its own process or even if the modules are distributed on different computers, the need for one or more middleware(s) arises. A middleware abstracts from the complexity of the components being connected to each other and provides an abstraction layer to communication technologies such as ethernet-based systems, buses or shared memory, in order to provide a common communication protocol, shared among the modules.

Robotics is a field that can hugely benefit from modular frameworks due to the variety of different technologies used. Various sub-domains that differ in many points (mobile platforms, industrial robots, aerial robots) rely on the same base technologies like vision, control and reasoning. Modular frameworks can therefor excel in robotics as they allow to reuse the base technologies. The most common modularization frameworks in robotics technologies are OROCOS [28] and ROS [104]. OROCOS is the older framework and follows a component-based approach where all components have to be loaded into the OROCOS deployer. ROS follows a more decentralized approach where modules have to use the ROS libraries to access common ROS infrastructure. This means that existing code in supported languages

such as Python or C++ can easily be integrated into a ROS system. This approach offers advantages in decentralization as ROS completely abstracts from the machines, that a so called ROS node is running on, but makes it hard to assure real-time behaviour. OROCOS on the other hand is harder to distribute among machines. It relies on CORBA for this task but requires every connected machine to run the OROCOS deployer. This approach however allows for hard real-time behavior when a single deployer, a single machine and a supported realtime operating system are used.

Both frameworks follow the open source principles and are highly community driven. This led to a integration of both frameworks, allowing developers to use the advantages of ROS and OROCOS in one and the same system. Newer approaches add the missing hard real-time capabilities to ROS if certain constraints are met [3]. Apart from ROS and OROCOS, other frameworks such as ROCK [60] that follow similar approaches are used in science.

Due to the modularization strategy, ROS is used in this work. The loose coupling approach is well aligned with the autonomously acting components that have to be coupled with each other. The name Robot Operating System (ROS) is misleading as ROS does not provide operating system capabilities. Instead of this ROS provides tools that allow to build modular robotic systems. ROS is supporting the major operating systems GNU/Linux, Windows and Mac OS and comes with an own build system. Machines running ROS can be controlled by other machines running ROS, using the SSH protocol. ROS is used by most of the leading universities, working in robotics, and an industrial version (ROS-I) is forming [6]. In 2010 more than 1000 ROS packages were available [2].

The basic principle of ROS is a decentralized communication structure based on Ethernet. Native support is built in for TCP and UDP connections but in general other communication channels can be used as well. Even though the communication is decentralized, ROS requires a master that is built-in to the *roscore* which has to run on a machine that is accessible by all clients. The *roscore* consists of the master component, the parameter server and a logging node called *rosout*. The master's functionality includes a registration service and a naming service. This allows for the aforementioned abstraction from the machines running the ROS node. Each node can connect to the master through the ROS client libraries and can register itself and its provided resources to the master. This allows all nodes to access the registered resources of other nodes without the need to have any information about the other nodes or their locations. This approach is based on the observer pattern and is depicted in Fig. 3.1. The parameter server of the *roscore* can be seen as a central database where parameter values can be stored by the nodes and where these parameter values can be accessed by other nodes. This is particularly useful for configuration purposes.

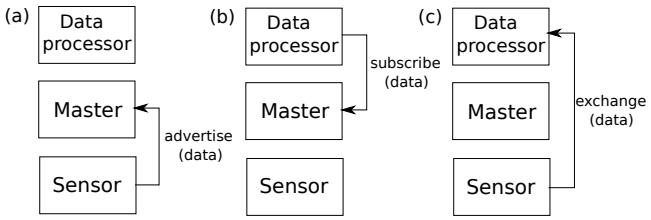


Figure 3.1.: The communication concept of ROS using the Observer pattern for the topic *data*. (a) a sensor node advertises a topic and registers it at the Master. (b) a data processing component requests to subscribe to the topic and retrieves the address from the Master. (c) The sensor node provides the data to the data processor using the topic *data*

A resource of a node can be of the type *Service* or *Topic*. In this scope a topic implements asynchronous one-to-one or one-to-many communication. Topics are an implementation of the publisher-subscriber pattern. This means that a node can advertise a topic and register this topic at the master. Every other node can subscribe to this topic via a query to the master. After the registration the master provides the address of the subscribing node to the advertising node and a direct connection between these nodes is established, which decentralizes the communication between the nodes. If a message on a topic is sent, the communication is handled asynchronously. The sender does not have to wait for an acknowledgment of the subscribers and data can be lost.

Services can be seen as Remote Procedure Calls (RPC) and implement synchronous one-to-one or many-to-one communication. The handshake between a service provider (server) and a service client (caller) is implemented in analogy to the handshake of a ROS topic. A Service in contrast to a topic notifies the caller about the answer of the server. This means that requests are confirmed by a response from the server. Both request and response can also be used to transmit payload from a node to another node.

Services and Topics are based on ROS messages. These are data formats for transmission, that can be defined by the programmer. Standard types are distributed together with ROS.

ROS also includes a feature called *dynamic_reconfigure*, which allows for a configuration of ROS-enabled nodes at runtime. For this purpose a node that supports dynamic reconfiguration exposes a set of its parameters (the reconfigurable node acts as a server) that can be accessed by client programs, such as the workflow based controller developed in this work or graphical user interfaces that allow to reconfigure a node.

Additionally, ROS includes functionality to examine transmission, to visualize

data, to navigate the ROS-related file system, to log, to handle threading and much more.

3.2. Computer vision

Computer vision is a basic technology for navigation and localization in robotics and is used in many other fields such as property protection, general automation and user interaction. Vision technologies can be divided into 2D visible light-based systems, 2D non-visible radiation-based systems and 3D technologies that can be based on both, visible and non-visible light technologies. 2D vision systems enable to identify objects, allow for basic navigation and tracking as well as applications demanding non-invasive temperature detection. 3D vision-based systems usually are more challenging to implement but facilitate to geometrically reconstruct the supervised scene without prior knowledge, allowing for volumetric measurements and 3D localization of known and unknown objects. This work is based on 3D vision technologies, whose basic principles are described in the following.

3.2.1. 3D vision

3D vision enables a camera, an animal, or a human to assess the position of objects and their movements in 3D space. Information about texture or shape can be derived additionally. Traditional 3D camera techniques are derived from 3D vision techniques in the animal world. Different principles are used, such as

- **Stereopsis:** A 3 dimensional scene is perceived from two different observation points at one and the same time. The 3D position of the objects can be measured via triangulation using the triangle composed of the viewpoints and the object itself.
- **Accomodation:** The lens system focuses on a point that is given attention to. The focus itself is a function of the distance between viewpoint and object. The focus can also be used in 3D computer vision depth measurement and the method is known as *depth from focus* [44].
- **Overlapping objects:** If an object is overlapping another object, the object has to be closer to the viewpoint than the object being overlapped.
- **Identification of objects with known size:** If an object of known size has to be detected, the distance and the position of the object in 3D can be estimated using the known focal length of a lens system and

the relative position and extend of the object with respect to the image acquisition system (sensor, retina).

- **Perspective:** If objects are known, the position of them in 3 dimensional space can be estimated using perspective such as the parallel edges that can appear to converge.

The human perception system is able to utilize additional principles to estimate the position of objects in 3D space. All described principles except stereopsis can be used with a single 2D camera system and optics. The principle of stereopsis is commonly used in 3D perception and is known as stereo vision. OpenCV [27] as highly popular and outstanding machine vision framework includes several of the above described 3D vision techniques. It allows for the 3D stereo calibration of two 2D cameras and the associated use as a stereoscopic camera system as well as the identification and localization of known objects such as checkerboards in 3D space based on perspective and object size. Common to all principles is the need of an intrinsic calibration of the optical system that is used. This is necessary due to an imperfect manufacturing process and unknown parameters of the lens systems. The intrinsic calibration allows for the partial estimation and compensation of these parameters and imperfections. This results in a higher measuring accuracy both in 2D and in 3D space. If the estimation of positions of objects with unknown dimensions is necessary for the application, a true 3D system or a 2.5D system such as a stereoscopic camera system is required. This condition applies both to measurements in highly unconstrained environments such as an operating room, as well as to the tracking of human motion in 3D. As soon as two or more optical devices are used in a camera system an extrinsic calibration has to be performed. The extrinsic calibration is a registration method that computes the angle and the distance between the optical devices.

3.2.2. 3D Camera principles

Several 3D camera techniques are known whereas triangulation based systems are most relevant to this work. These techniques include:

- **Plenoptic cameras [11]:** This principle uses a microlens array in contrast to traditional optical systems allowing the sensor to capture a 4 dimensional lightfield. The additional 2 dimensions (in contrast to 2D cameras) represent the directions of the lightrays that were measured with the sensor array. This additional information allows for refocusing and therefor for the estimation of the position of an object in 3D space.
- **Time-of-flight cameras [43]:** Pulsed light, often in the infrared-spectrum, is transmitted using a light source. The light is reflected from

objects that are in the field of view of the camera. The reflected light is measured using the sensor array of the camera. In order to avoid stray radiation a filter in front of the sensor can be used. The elapsed time from transmission until the reflected light is measured and the distance of the objects from the camera can be computed using the known speed of light.

- **Structured light cameras [110]**: A projector is extrinsically registered to a camera. A known pattern is projected onto the scene and the captured structured pattern is analyzed (e.g. its distortion) in order to find the 3D points corresponding with the object.
- **Triangulation-based systems [52]:** Requires at least two cameras (or a camera and a projector) to acquire a 3D representation of the image. The principle is based on the different viewports of the cameras, the known distance (baseline b) and angle between the cameras. The 3D coordinate of a point can be computed using this parameters together with the locations of the perceived point in both of the synchronized images of the two cameras. The light ray from the object through the lens creates a signal on the sensor. The direction of this ray can be computed if the intrinsic camera parameters are known. If the extrinsic camera parameters are known (baseline and angles between cameras) and the object is in the field of view of both cameras, one can compute the rays for the object in both camera coordinate systems. The 3D coordinate can be found geometrically and is located at the position where the two rays of the cameras cross each other. The basic geometrical principles are depicted in Fig. 3.2. Triangulation is a computational expensive method as an object has to be identified in both cameras based on its 2D features. The used algorithms are usually influenced by lighting conditions.

In this work the Microsoft Kinect™ 360 and the Kinect™ One cameras are used. The Kinect™ 360 is a triangulation-based system that uses active illumination in the infrared spectral range. The Kinect™ 360 uses an infrared laser projector that projects a pattern of dots. This pattern is analyzed using the infrared camera built into the Kinect™ 360 camera, the known baseline and stereo algorithms. In order to further improve the accuracy of the depth measurement, an astigmatic lens is used in front of the projector. As in an astigmatic lens the focal length in horizontal and vertical direction varies, the dots projected become ellipses, whose orientation and major- as well as minor-axis are dependent on depth and position with respect to the projector [115] [116]. This deformation is analyzed using the infrared camera. This can be compared to the *depth from focus* method. Both methods combined deliver a high-quality depth map with a resolution of 640x480 pixels and 11 bit depth. In terms of completeness of the point cloud the approach is favorable

compared to most conventional stereo camera technologies. Humans are not affected by the system in their visual perception due to the infrared illumination that is not in the visible spectral range for humans. However, as active illumination is used, the approach is prone to interference when strong infrared light sources are present. This results in increased presence of noise when multiple Kinect™ 360 are used. Also outdoor usage is only possible to a very limited extend due to the infrared radiation of the sun.

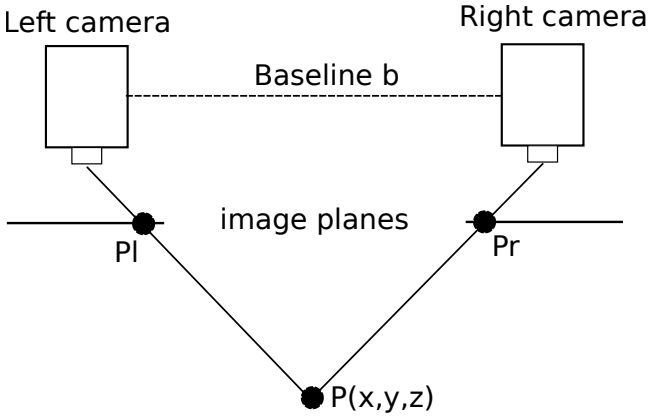


Figure 3.2.: The principle of triangulation-based systems using an angle of 0 degree between the cameras and a known baseline b. P is the point observed in 3D, P_l and P_r represent the locations on the camera sensors where the reflected light from the object observed generates the signals.

The Kinect™ One is a Tof system with a resolution of 512x424 pixels [103]. The depth data quality is superior to the one of the Kinect™ 360. A cause for this is that the Kinect™ 360 projects a low resolution pattern of dots. The depth for parts of the surfaces, on which the dots are projected on, that lie between two or more dots can not be measured and has to be interpolated from the measured depth at the locations of the dots in the neighborhood. In contrast to this, the Kinect™ One is able to measure the depth for each of the pixels of the sensor array.

3.3. Workflow Management

Industrial processes nowadays are massively driven by business processes. These processes can e.g. affect delivery, production and financial resources. A business process is repeatable and its aim is to help in the value creation process [140]. An important topic in the scope of business processes is the process management, which aims to create and optimize business processes

based on the companies strategy, resources, customer satisfaction or customer needs. Enterprise-Resource-Planning(ERP) is the management of resources of a company that performs ERP. ERP helps to improve the value creation process and is therefor well aligned with process management. ERP Software products such as SAP ERP (SAP, Walldorf, Germany) often include a module that can handle business processes.

The creation and maintenance of a business process is different for most companies and depends on the available resources of the company. The process itself should be carefully modelled to optimize and track the effectiveness of a process and to optimize the value creation process. Several workflow management systems are available that help for the definition of the processes (graphical or in a programming language), for the execution of the processes and that allow for the implementation of interfaces that enable to execute atomic tasks of a process (such as part assembly) on the companies resources (such as machines).

Some of the major business process modelling concepts that are used in the workflow management systems are described in the following:

- **BPEL:** The Business Process Execution Language is a programming language that is based on XML. It is not designed for a graphical definition of the business processes and its core concept is to implement activities to be performed using webservices. This allows for a decentralised execution of activities in a heterogeneous system.

- **BPMN:** Business Process Model and Notation is a graphical programming language for the specification of business processes and workflows. BPMN is influenced by sequential concepts from languages such as UML (activity diagrams). The basic concept of BPMN is to have flow objects that can e.g. represent an activity of the business process and connecting objects, which allow for the connection of flow objects. Compared to BPEL or XPDL, BPNM is not an execution language and processes in BPMN usually need to be translated to an executable language before execution. An option is the translation to XPDL or BPEL that allow for a direct execution. An example for a process in BPMN is shown in Fig.3.3:

- **XPDL:** The XML Process Definition Language is an XML-based language like BPEL. It is strongly driven by the WfMC (Workflow Management Coalition) and follows the Workflow Reference Model of the WfMC [56]. XPDL is designed for the support of BPMN models and allows for its representation whereas BPEL is not designed with respect for the support of BPMN and is more focused on the activities that are connected to resources via webservices.

The three concepts are converging due to the efforts of the WfMC. The described concepts are usually integrated into complete business process modelling and execution or workflow modelling and execution suites such as IBM BPM-SOFTWARE (IBM, Armonk, NY, USA).

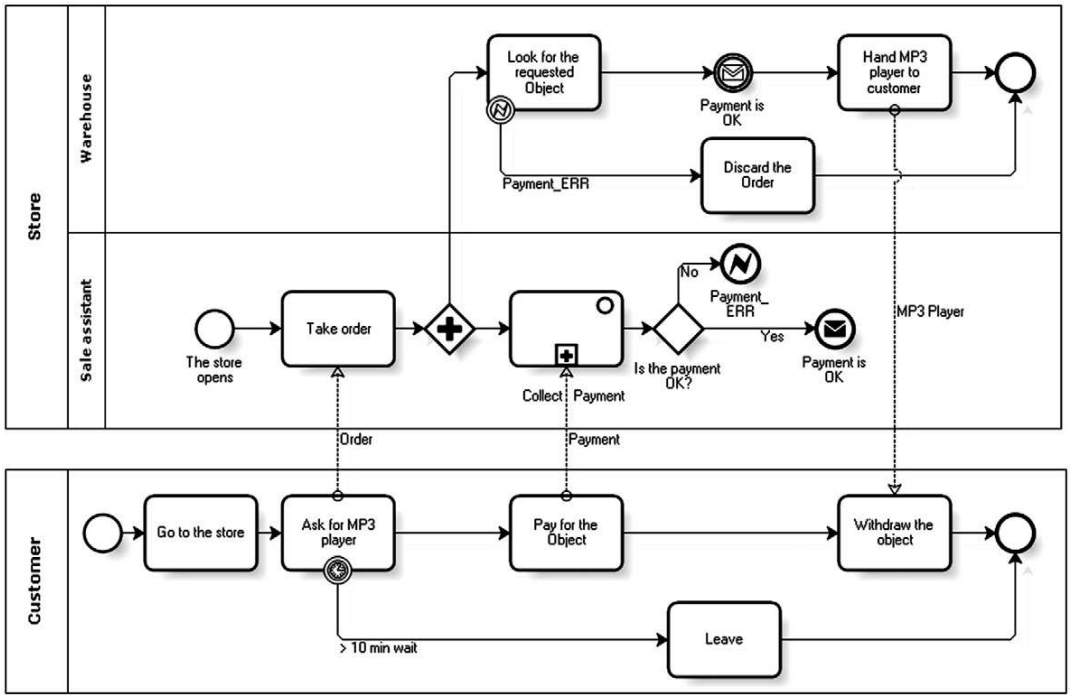


Figure 3.3.: A simple process in BPMN, source: [33], ©[2011] Elsevier B.V.

3.3.1. WfMC

The Workflow Management Coalition was founded in 1993 and aims to create standards in the field of workflow management and business process management [8]. Its major contributions are the Workflow Reference Model and the XPDL language. Members in the WfMC are companies working in the field of business process management, companies that adopt BPM solutions, developers and many more. The WfMC reference model (WfMC-TC-1003) [56] helps to standardize complex systems. It describes the components of a system and how they can be connected and interfaced to each other. The reference model is vendor-independent and built around a Workflow Enactment Service.

The Workflow Enactment Service is defined as:

> A software service that may consist of one or more workflow engines in order to create, manage and execute workflow in-

> stances. Applications may interface to this service via the workflow application programming interface (WAPI) (Workflow Management Coalition The Workflow Reference Model, source: [56], p.21, ©1993, 1994, 1995 The Workflow Management Coalition)

A Workflow engine is defined as:

> A software service or "engine" that provides the run time execution environment for a workflow instance (Workflow Management Coalition The Workflow Reference Model, source: [56], p.22, ©1993, 1994, 1995 The Workflow Management Coalition)

Finally a workflow itself is defined as:

> The computerised facilitation or automation of a business process, on whole or part (Workflow Management Coalition The Workflow Reference Model. source: [56], p.6, ©1993, 1994, 1995 The Workflow Management Coalition)

Also important for this work is the topic of worklists and worklist handlers. A worklist is a list where items, activated by the workflow enactment service, are placed to be processed by worklist handlers. Worklists are created by the engine from a process and its tasks. An item in this scope can be anything that needs to be externally processed. A worklist handler is the module of the application that is responsible for the processing of the items that are placed in the worklist for the specific worklist handler. This can be e.g. a desktop application or a mobile application that interfaces the user to the workflow enactment service, or more general a client application that is involved in the workflow execution process. The reference model involves the use of five interfaces depicted in Fig.3.4, that allow to connect to the Workflow Enactment Service and that are described below:

1. **Process Definition Tools:** This interface is used to exchange process definitions or parts of them between the workflow modelling tool and the Workflow Enactment Service.

2. **Workflow Client Applications:** This interface connects the participants of a workflow to the Workflow Enactment Service using the previously depicted concept of worklists and the worklist handler. Additionally, this interface can be used to control processes, such as the creation or the termination of an instance of a process, to supervise or administrate processes but also to exchange data between the enactment service and the client application.

3. **Invoked Applications:** This interface is intended to be used to invoke external applications that cannot use the standardized interface 2. Such applications can communicate via their interfaces with an application

agent that is connected to the Workflow Enactment Service through the standardized API (WAPI).

4. **Other Workflow Enactment Service(s):** This interface is particularly useful if more then one workflow management system is used in the application. The interface 4 provides standardized mechanisms to connect different workflow management systems to each other to allow for a distributed execution of the processes involved.

5. **Administration & Monitoring Tools:** Interface 5 facilitates to administer the workflow management system. Through Interface 5, users or user groups can be managed and user roles can be defined. Also, resources and the process itself can be supervised or information about a process or process instances can be acquired.

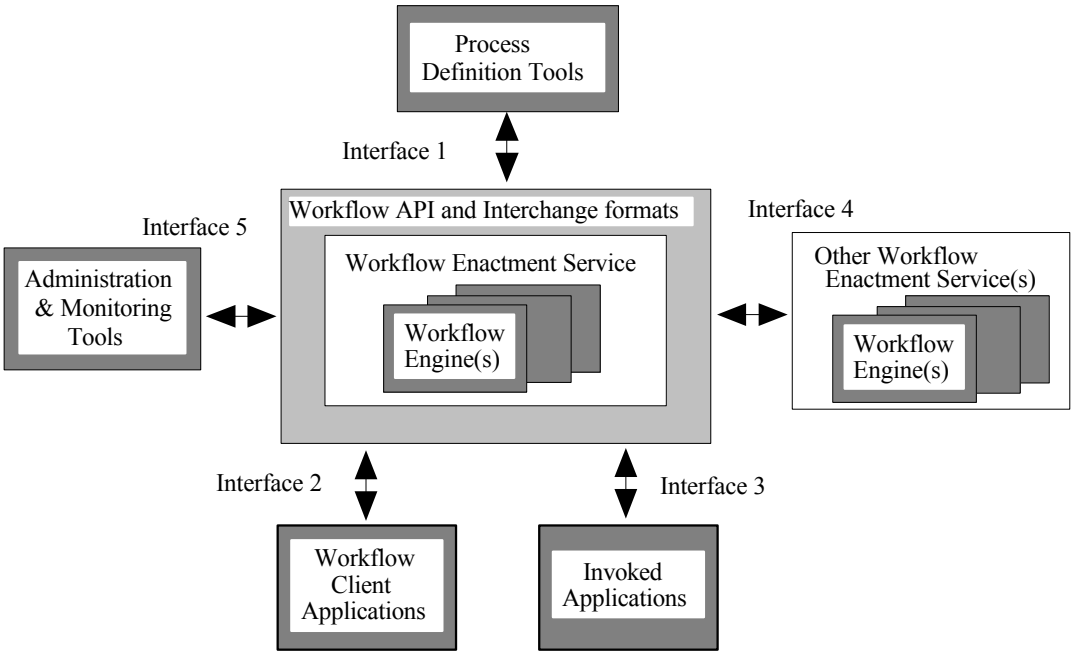


Figure 3.4.: The components and interfaces of the WfMC Workflow Reference Model source: [56], ©[1993, 1994, 1995] The Workflow Management Coalition

Additionally, the WfMC Reference Model gives common definition about system states, different data types to be handled by the system components, allows for role, user and resource management and defines expressions in the scope of workflow management to allow for a common terminology among the members of the WfMC. Workflow management systems that follow the reference model and implement the WAPI are intended to be easy to interface to other systems, or client applications that also follow the WAPI.

3.3.2. YAWL

Yet Another Workflow Language (YAWL) is a workflow management system that follows the principles of the reference model of the WfMC. YAWL is a complete workflow managment system with multiuser support, support for roles and its language has a high expressiveness. BPMN Models can be directly mapped to YAWL models. YAWL is entirely JAVA based. The language does not entirely follow the standardization efforts of the WfMC but is completely open source and is based on well-established open source technologies such as XML, XML-Schema, XPath or XQuery. It is based on Reset-Nets as well as on their underlying Petri-Nets and supports most workflow patterns [129]. It is designed as a Service oriented architecture (SOA).

The YAWL engine as execution and core component of the system runs on an Apache Tomcat Server. Administrating components such as the resource service are services that run in the same server container and can communicate with the YAWL engine. The YAWL editor, which is also JAVA-based, is a graphical programming and definition tool for YAWL schemes that can be uploaded to the YAWL engine via the resource service. A YAWL scheme is a stateless definition of a workflow that can be instantiated by the YAWL engine on request.

The editor can be dynamically connected to the YAWL engine to acquire information about users, resources or user roles. All available services that are registered with the YAWL engine are known to the engine and the editor to allow for an association of tasks with services. A service in the scope of the SOA can e.g. be the resource service or a custom service that is connected to the engine to allow for external application support through the interfaces 2&3 of the WfMC reference model. In the scope of this work, a custom service has been designed that allows for the connection of YAWL to the decision making module described in Chapter 5.2.

Due to its open nature YAWL offers easy extensibility. There are several projects in the scope of workflow modeling, which support YAWL and that allow to improve the capabilities of workflow engines in order to e.g. allow to perform process mining. Process mining allows to gain knowledge from the analysis of the process execution [130].

3.4. Reasoning

Reasoning is a topic that is closely connected to logic and has already been described in the antics, where Aristotle and Platon dealt with consciousness,

decision making processes and problem solving. Reasoning can be described as a process that uses new or existing knowledge to help to justify actions, to verify or to evaluate statements or to find causalities. It can be associated with learning, intelligence or consciousness. In computer science reasoning is crucial to artificial intelligence (AI), especially in classification or other knowledge-based systems. It is clear that in terms of state-of-the-art artificial intelligence, reasoning does not include conscious actions. Logic is the underlying principle to most if not all decision making systems in information technology. Logic describes how to make conclusions based on a reasoning process. Logic can be divided into deductive reasoning, inductive reasoning and abductive reasoning [17].

Deductive reasoning means that a certain conclusion can be found using existing statements like

- Patients are harmed during surgical procedures
- Prostatectomy is a surgical procedure

Deductive reasoning allows to find the conclusion

- Patients are harmed during prostatectomy

Inductive reasoning allows to find evidence for the truth of a conclusion based on existing statements such as

- Patients are harmed during all known surgical procedures

Inductive reasoning facilitates to find evidence that a newly found procedure is probably harmful for the patient.

Abductive reasoning means that an explanation for one or more observations is found based on the knowledge of the system that performs abductive reasoning such as

- A Patient was harmed during a surgical procedure
- Prostatectomy is harmful to patients

Abductive reasoning allows to conclude that the harmed patient underwent a prostatectomy based on this knowledge (because there is only one procedure defined).

In this example only inductive reasoning can provide a certain conclusion.

Logic can further be divided in bivalent classical logic including propositional and predicate logic and in non-classical logic such as multivalent logic.

3.4.1. Descriptive logic and ontologies

Descriptive logic is frequently used for knowledge representation. Knowledge from the real world can be formalized to be represented in description languages. Most descriptive logic languages are closely related to first-order predicate-logic and often share its decidability. Description logic is commonly used in knowledge-based information systems in order to allow for decisions or to explain relationships. Descriptive knowledge gained great popularity in the scope of the semantic web, which can be seen as an extension to the world wide web and allows to represent domain knowledge in a machine readable form. Two key concepts for knowledge representation, TBox (terminological formalism) and ABox (assertional formalism) are defined. TBoxes describe intentional knowledge as a terminology and therefor general concepts of the knowledge base. ABoxes describe extensional knowledge. This means that TBoxes intend to describe knowledge that is not subject to change and describes the relationship between concepts in the knowledge-base. ABoxes describes knowledge that can be subject to change and especially knowledge about the individuals is described [17]. In this scope the W3C (World wide web consortium) [9] released the OWL standard. This standard describes the web ontology language with the different available language levels [83]

- **OWL-Lite:** OWL-Lite is a subset of OWL-DL that is designed to create a language that can help to formalize knowledge and that is easy to implement.
- **OWL-DL:** OWL-DL is equivalent to first order predicate logic. A variety of automated reasoning systems provide support in order to reason on knowledge that has been implemented in OWL-DL.
- **OWL-Full:** Has less restrictions that OWL-DL and is inspired by higher order predicate logic and therefor reasoning can be limited to the OWL-DL subset included in OWL-Full.

Knowledge bases are frequently described as ontologies. An ontology in the information technologies context is a knowledge base that is intended to provide domain knowledge to computer systems that deal with knowledge such as systems that are supported by artificial intelligence. A properly designed ontology has to assure its integrity and should allow for inference through reasoners. As a standard that is easy to implement and well accepted in the scope of the semantic web, OWL has been well established as a language to define ontologies. In order to design these knowledge bases, OWL utilizes classes, properties and individuals that are described in the following.

- **Classes:** Define of what type an individual can be. e.g. human, animal or plant.

- **Properties:** Define the relations between classes, individuals and data. e.g. *Human eats Plant or Animal*.
- **Individual:** Individuals are instances of classes. e.g. *Aristotle is a Human*.

These design principles OWL are fully compatible to TBox and ABox concepts of descriptive logic.

3.4.2. Probabilistic and Rule based reasoning approaches

As described in the previous chapters, reasoning is the process of taking conclusions or to explain decisions. Many of the common reasoning processes in information technology are based on probabilistic approaches, e.g. based on the Bayes rule, or on rule-based approaches that use knowledge from e.g. ontologies. Combined approaches are possible as well. Probabilistic approaches are used for classification, clustering or learning. Common approaches include classical Bayes networks, Hidden Markov Models, Random Decision Forests, Neural Networks and many more. In the scope of knowledge-bases, probabilistic approaches can be used as well. An example are MEBN (Multi-Entity Bayesian Networks), where knowledge can be stored in a knowledge-base together with probabilities to allow for probabilistic reasoning in the knowledge base [75]. More frequent in terms of knowledge processing in ontologies are rule-based reasoning systems, that are implemented for OWL in systems such as FACT++ [127], [118] or HermiT [113]. Such systems utilize forward-chaining, backward-chaining or tableau [106] and hypertableau [87] -based approaches. Classical probabilistic methods usually require training and allow for situation detection or object classification problems. Huge knowledge bases are commonly utilized for domains where a huge amount of knowledge has to be formalized and integrated such as decision support systems e.g. for therapeutic purposes in medicine. Rule-based reasoning however does not necessarily require a huge database. Simple approaches can have their foundations on simple conditional sentences. As used in MEBN, rule-based reasoning and probabilistic reasoning can be combined to merge the advantages of both approaches.

4. Concept and Design

The following chapter shows the proposed approach of a control concept for integrated operating theatres that are equipped with one or more robots. In the latter chapters, the components of the system shown in the following are described in detail.

Existing systems for surgical operating theatres or surgical robots are mostly single-vendor solutions that constrain the hospital and the surgical personnel to the products and components offered by this vendor. A solution to integrate components, provided by different companies, can be implemented through the definition of standards or by using communication servers. Both concepts are already used in Hospital Information Systems (HIS), which are usually based on HL7 and DICOM and that interface between and integrate the information from different subsystems, included to the HIS. An approach to transfer the concepts to the operating room is followed by the OR.NET project [70]. Several questions however remain open. Integration allows the subsystems to control other subsystems or to gather data/information from them, but often it is not defined how the systems have to behave and to interact, which gives away potential.

It is crucial to understand the needs of the surgical staff and to translate them to appropriate and optimized system behavior. Machine learning methods, such as Random Decision Forests (RDF) can be used to classify sensor data and to judge about the personnel's needs. Due to the complexity of surgical operations and highly variable actions during procedures, traditional machine learning methods can be hard to implement, are error-prone and limited to a subset of tasks, or require to invade the procedure.

The proposed approach is based on workflows and breaks down the task recognition within a workflow and the system control tasks into smaller tasks that can be handled using conventional methods. It allows for easy integration of different operating room components, workflow driven operating room configuration and to adapt the operating room according to the state of the workflow as well as to the needs of the personnel. This will eventually lead to a *smarter* operating theatre, which adapts to the surgical tasks.

The control system itself is composed of three main parts that are entirely based upon ROS.

- The workflow-based controller is the central component of the system. It uses a predefined workflow for the planned procedure and dynamically selects, links and configures the system components for each task, based on the requirements for a task, the availability of the components and the compatibilities of the components. The controller follows principles used in industry and adapts them to the heterogeneous procedures in the operating room .
- The perception system serves the purpose to provide input data to follow the workflow-based controller in a non-disruptive and non-invasive way. The system uses a multi camera approach, which is based on RGB-D cameras, to acquire a high-resolution 3D representation of the current configuration of the operating room. Compared to conventional methods, there is no need to introduce additional equipment to the sterile areas of the operating room, such as optical markers or sensors.The geometry of the camera configuration allows to place the cameras outside of the laminar airflow within the operating room, avoids occlusions and does not disrupt the personnel in any way. The system's detection capabilities are focused on the actions and movements of the personnel and can be complemented by additional conventional sensors for instrument or patient-centered observations.
- The execution components to be used during the intervention itself can be of various natures. In the proposed scenario, a telemanipulation system with autonomous and cooperative features is focused on. The components of this system include lightweight robots, components that can give feedback to the staff (e.g. visualization) , input devices, sensors and software components, like drivers or algorithms.

In order to achieve the goal of an optimal system composition and configuration for surgical assistance using workflow tracking, four paradigms have been used for the design and implementation of the system:

- **Workflow based process execution:** The concept has been adapted from industrial manufacturing or delivering processes and allows to break down a big process into small sub tasks.
- **Knowledge based system composition:** An ontology is used to allow for the composition and configuration of the system for a task invoked during the workflow execution. The ontology is comprised of task and component knowledge.
- **Configuration instead of programming:** In order to decrease the effort for integration of new components and to easily use them to enhance the system performance, a configuration based approach is used. Already available components, or components that do not necessarily have to be

designed for the shown system, can be modelled in the knowledge base with their interfaces and requirements and can then be immediately used without programming effort. The presented implementation uses only software components, which are implemented as ROS nodes.

- **Sensor-based process supervision:** A rich sensor system is used in order to gather the necessary data for process supervision. Its data is interpreted using the system, which is composed by the workflow-based controller during runtime, in order to notify the workflow-based controller about finished tasks of a workflow.

In combination the concepts aim to reduce the complexity of modeling and executing surgical processes in a computer based assistance system by the use of modularization. This includes the reduction of the complexity of the components to monitor the workflow state, which are often computationally expensive probabilistic models that need to be trained for one specific workflow [99]. In addition, the modularization allows for the reuse of components for different or evolving workflows.

The *Configuration instead of programming* approach allows to easily integrate new components and to test their behavior, effects on and benefits for the workflow. Reasoning support in the knowledge base facilitates to identify equivalent components in the system, to allow for a dynamic system composition in case resources are not available or resources require additional resources. The workflow-based process integration and its foundation in process modelling allows for the graphical definition of the workflow. The perception system delivers the input data for the recognition of finished task executions within a workflow.

4.1. Modularity through ROS

ROS has been chosen as base for the developed system due to its widespread use in robotics research, its modular approach and its built in mechanisms to externally control each subsystem. The components used for the task execution and for the operating room supervision are designed as ROS nodes. They follow the design principles of ROS where small subsystems communicate via topics, services and parameters. All of these components of the system apart from the *roscore* and the workflow-related components are referred to as execution components, which means that also the components of the perception system are are execution components. An execution component in this scope is a component, which can be used as a part of a system in order to collectively execute the task. The execution components of the system are

managed, configured and controlled in a generic way by the workflow-based controller. An overview about the entire system is shown in Fig. 4.1.

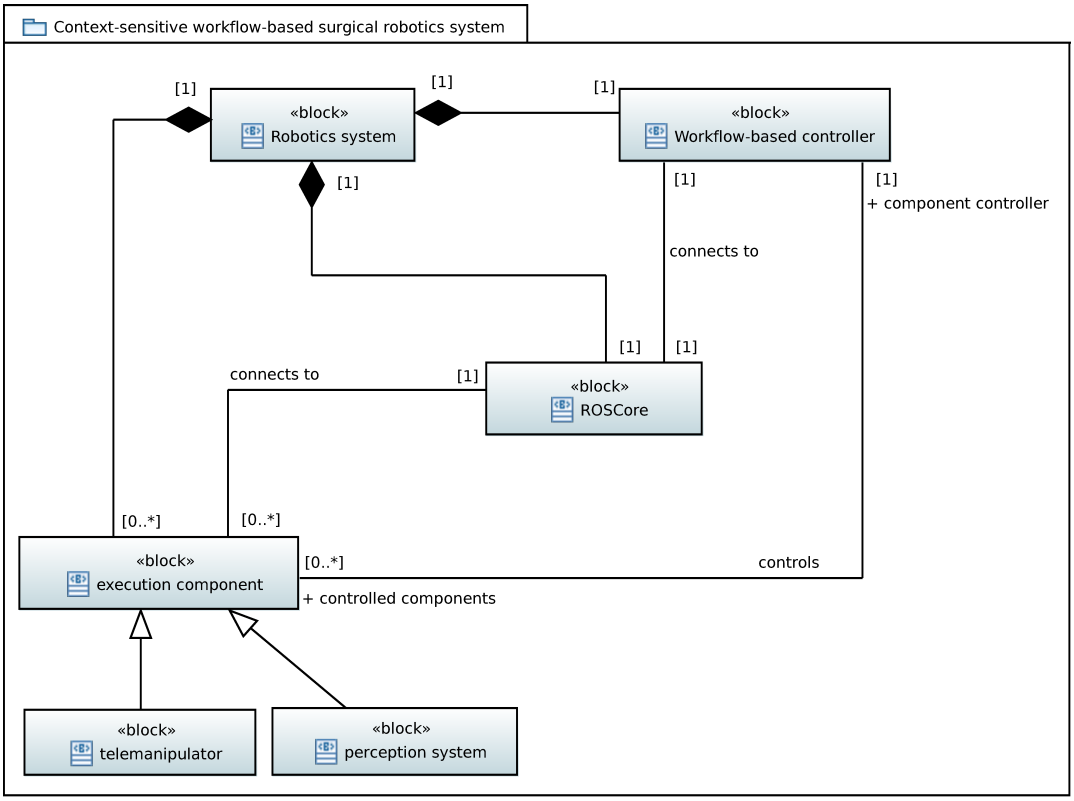


Figure 4.1.: SysML Block Definition Diagram showing the entire robot system composed of the roscore, the workflow-based controller and the execution components.

It can be seen that all components of the system are part of the Robotics system, whilst the workflow-based controller acts as the core of the system and the *roscore* acts as a management component for the ROS-based system parts. It has been assumed that all components within the system are designed as ROS components, to be controllable via the workflow-based controller. All components are execution components and subsystems as the telemanipulator or the perception system are composed of one or more execution components.

Special to the approach is that the subsystems, activated by the workflow based controller, remain operational even if the workflow-based controller fails. This is due to the design approach that the workflow based controller is designed as an add-on to ROS and uses existing mechanisms of ROS, meaning that an active configuration of a ROS system does not require the workflow-based controller. The workflow-based controller is never part of the system, which actually assists the personnel during the procedure, but

composes and configures the system at the beginning of every task of the procedure and detects the end of a task of the procedure. This means that after a potential failure of the workflow-based controller, a new workflow-based system composition is not possible and the current state of the system has to be used for the transformation of the system to a safe state.

4.2. Workflow-based controller

The workflow-based controller as core component of the control system is intended to allow for the generic modeling and the execution of surgical workflows. It has been designed to be driven by a priori knowledge about the system and the workflow as well as by the supervised state of the surgical process. It includes design principles from business process modeling to allow for data mining and graphical process definition by the expert. Additionally, in contrast to conventional process driven systems, where a process engineer defines a process and the engineering team implements the application using the process, a knowledge-based system is placed between the process execution system and the execution components to allow for an automatic system composition, based on the current operating room and personnel needs. This serves the purpose to allow for situation-adapted system behavior and flexible integration of new components or new tasks to the process . Additional attention has been put on straightforward remodeling of the process to allow for an iterative improvement of the process by the expert without the need of additional development efforts. The concept of the workflow-based controller concept that allows to realize these concepts is described in the following.

From a control theory's point of view, the execution of workflows, the supervision of the workflow and the interpretation of the data, which is fed back to the workflow-based controller, is similar to a closed loop controller. This is depicted in Fig. 4.2, where the basic control loop for workflow execution and tracking is shown.

The controller is primarily driven by the execution of a workflow by the workflow engine, which is part of the workflow-based controller. Using the knowledge from the knowledge base, which is also part of the workflow-based controller, about the task to be executed as part of a workflow, the workflow-based controller configures the system to be in the correct state to execute the requested task.

The workflow-based controller also selects and configures one or more suitable sensor systems to supervise the state of the workflow, which is performed via the detection of the end of an executed task. The output data from the chosen sensor system is fed into one or more situation interpretation modules,

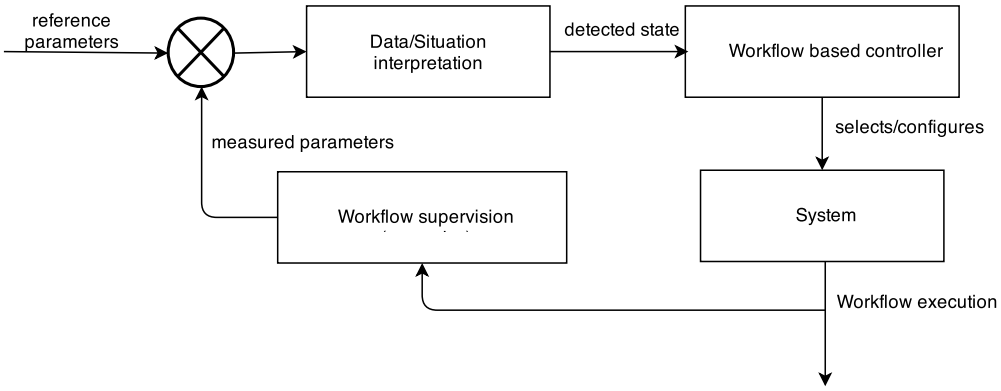


Figure 4.2.: The closed loop controller like design of workflow execution and supervision, including the workflow-based controller, the execution system (execution components), the workflow supervision and the situation interpretation as input for the workflow based controller.

which are also selected and configured by the workflow-based controller, based on the characteristic data for the end of the task to be executed.

Finally, the output data of the interpretation modules is fed back to the workflow-based controller, which includes a comparator to compare the situation detection modules' output with a reference value from the knowledge base. This is performed in order to judge whether a task has to be finished, or whether it still has to be executed. When the data from the interpretation module matches the reference value the task end has been detected. When the data from a sensor can be directly compared to a reference value, which is characteristic for the end of a task, the workflow-based controller can bypass any interpretation module and can directly compare the sensor's output to the reference value. As soon as a task is finished, the workflow engine activates the next task(s), if there are subsequent tasks in the workflow, and the workflow-based controller configures a new system configuration based on the new tasks' requirements.

It has to be mentioned that the workflow based controller internally does not distinguish between the components that execute the workflow, the sensor system to be used and the data interpretation modules. These components are treated generically as components, that are required to execute a task of a workflow and fall in the category of execution components.

Compared to a traditional controller where the reference value and a measured value are subtracted and the resulting error is translated into a control variable to update the system, the proposed controller uses an aforementioned

comparator. This means that the data interpretation modules or sensors serve as execution components that deliver control variables to the workflow-based controller, which are used as input to the comparator, in order to judge whether the control variable indicates the end of a Task or not.

Finally, the workflow based controller dynamically changes the structure of the control system. This means that the control loop is composed based on the tasks' requirements. This includes the exchange of the execution components, including sensors and data interpretation modules, as well as the exchange of reference values, and the reconfiguration of the internal comparator(s) to adapt it to the used data interpretation modules and sensors.

It may also be possible that several of the control loops are active for one task as there might be more than one criteria that indicates the end of a task. This happens if the workflow contains alternative branches in between which the engine has to select after the end of a task. In theory it is possible that multiple tasks are running simultaneously, which additionally affects the controllers configuration and that can require multiple comparators and multiple control loops.

The workflow-based controller utilizes the YAWL workflow language and consists of two ROS nodes, whereas one node, which in the following is called *YAWL Connector*, is in charge of interfacing the Apache Tomcat-based YAWL engine to the workflow-based controller, which is implemented using ROS. The second component, which in the following is called *Master Control*, is driven through the information from the executed workflow, an integrated OWL knowledge base and the entire systems current state. It acts as a centralized component, which is a controller for the decentralized control systems. The *Master Control* is capable of querying the currently available ROS nodes using the ROS Master API and allows to reconfigure ROS nodes during run time. This allows to dynamically connect and disconnect components and to configure them in order to adapt to the current situation. Fig. 4.3 shows the internal connections between the components of the workflow management system.

The workflows are usually generated by a human operator and are stored in the YAWL engine The components of the process are also stored in the knowledge base. The knowledge base is a crucial component of the system. It does not only include knowledge about the possible tasks and available components but does also include knowledge about the relations between components and components, tasks and components and tasks and tasks. The internal structure and the relations defined in the knowledge base are described in detail in chapter 5.2.3. The knowledge from the knowledge base has to be designed carefully and has to be correct at any time as it defines how the system is composed, how the data/information is distributed among the components and is used within the Master Control.

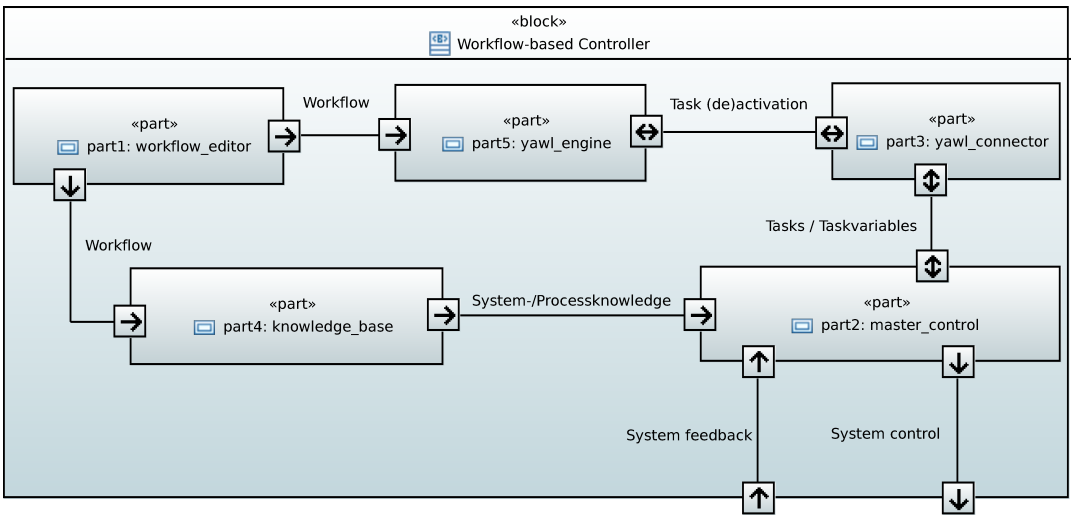


Figure 4.3.: SysML Internal Block Definition Diagram showing the internal connections of the workflow management system.

Using the knowledge base, the master control can compute all possible future states of a workflow to allow for the computation of suitable system configurations. The task invocation is controlled by a workflow, which is executed in the YAWL engine. As soon as a Task is getting started by the engine, the Master Control is notified. The YAWL engine utilizes the YAWL Connector to exchange information with the Master Control, using ROS. The Master Control is connected to the knowledge base to gain information about the process and the system and communicates with the YAWL engine using the YAWL connector. Using this architecture, the YAWL engine controls the entire system, while the Master Control is in charge of the correct configuration and setup of the execution components. As shown before, the Master Control also notifies the engine, as soon as a task has finished, to allow for the activation of subsequent Tasks.

4.3. Perception System

In order to allow for the tracking of workflows, suitable sensor systems are required. As this work focuses on the operating room staff's workflow and environment, an environment perception system is required to allow to track humans and to provide features for workflow detection methods. Due to the non-invasiveness of optical systems, the wide field of view and the wide range of detectable objects as well as actions, a camera-based approach has been chosen.

The perception system is designed to allow for a complete environment

supervision of the operating theatre. The main criteria for the design of the system were:

- Redundancy to avoid occlusions
- Stable people tracking and detection
- Maximization of distance to sterile parts of the operating room

Additional emphasis was put on the ability to completely represent the operating room scene, including object positions and a history for trajectory analysis, to allow for the extraction of movements of the objects, such as human and robots.

The systems design includes completely markerless tracking methods to allow for the detection and segmentation of known objects or humans from the scene without the need of optical trackers. This allows for a higher flexibility compared to marker-based systems. Using a markerless approach, the system can react to unknown objects that are not equipped with markers.

An investigation of available camera techniques revealed three key technologies that are suitable for the design of such a system, namely Time-of-flight systems, stereo-based systems and lightfield systems. Although the system is designed to work with all techniques, the original design has been implemented using the Microsoft Kinect™ 360 which can be classified as a stereo-based system with active illumination. This decision has been taken due to the target application of human detection and tracking for which the Kinect™ 360 is designed and where it surpasses other existing systems.

Compared to passive stereo systems, the Kinect™ 360 delivers a complete point cloud. Additionally, the 3D depth map is created internally and therefor external high performance computers are not required for depth image acquisition. Compared to contemporary Time-of-flight cameras, the Kinect™ 360 delivers a resolution that is more than 7 times higher, which allows to extract finer details from the scene and which enables algorithms to allow for precise skeleton tracking of human bodies. Lightfield cameras offer high potential for precise 3D point cloud generation, but if high resolutions are desired a massive amount of computation hardware is required and the camera system is expensive compared to both, time-of-flight and stereo-based systems.

With the Kinect™ One camera [103] announced in 2014, Microsoft changed the working principle of the Kinect™ camera family from stereo-based to time-of-flight, using a high resolution ToF sensor that delivers 512*424 pixels sized depth images. This resolution is nearly as high as the resolution of the Kinect™ 360 (640*480). The Kinect™ 360 measures depth values at predetermined positions in the camera images and computes intermediate depth values in between them. The sensor array of the Kinect™ One is able to measure depth values for every pixel of the sensor, which removes the need

of unprecise intermediate depth interpolation. For the application, the more precise depth information of the Kinect™ One is beneficial, which surpasses the drawback of a lower depth image resolution. The methods from the original Kinect™ 360 system have therefore been applied to the Kinect™ One. Both systems have been evaluated in terms of registration accuracy, tracking and detection quality.

The camera positions for both systems have been chosen to allow for a frontal body tracking of the personnel in the operating room, to allow for mounting positions outside of the laminar flow in the operating room and as far away as possible from the patient to reduce the risk of introducing unsterile objects to the operating field. The desire to take frontal images of the humans is motivated by the need to track the position of the hands of the surgical personnel, which are in the operating room usually located in front of the bodies. The challenge in this scope was to find positions where the system is impassible to occlusions with respect to the personnel's most relevant positions in the operating room. Additionally desired redundancy to allow for continuous 3D capturing of relevant objects and humans even in the presence of an occluded camera, reduced the set of suitable camera positions.

These requirements resulted in a four camera square shaped configuration, which is ceiling mounted. The angle between the cameras' optical axis and the floor has been chosen to be as accute as possible in order to get the desired frontal view of the humans in the operating room. The camera orientations were chosen manually in a way that the optical axis of all cameras are targeted to a point slightly above the center of the operating table. This position is usually close to the situs where the most relevant actions during a surgical procedure take place. The chosen configuration allows to track people standing along the long sides of the operating table, using at least two cameras.

Inherent to the operating room setting are occlusions of the lower body parts of the personnel, which are introduced by the operating table an cannot be avoided using a camera configuration for the frontal view of the humans as can be seen in Fig. 4.4. Due to the low relevance of the lower body configuration of the operating room personnel, this issue has not been addressed in this work. A view of the final camera configuration, which is similar in shape for both camera systems, is shown in Fig. 4.5.

Additional challenges were introduced by possible interferences when multiple active camera systems are used. In fact, the Kinect™ 360 cameras interfere with each other as can be seen in the work of Faion et al. [40]. This results in an increased noise with every additional camera added. No interferences between a Microsoft Kinect™ 360 and a Microsoft Kinect™ One could have been observed during the experiments. However, the use of multiple Kinect™ One cameras revealed interferences. These interferences

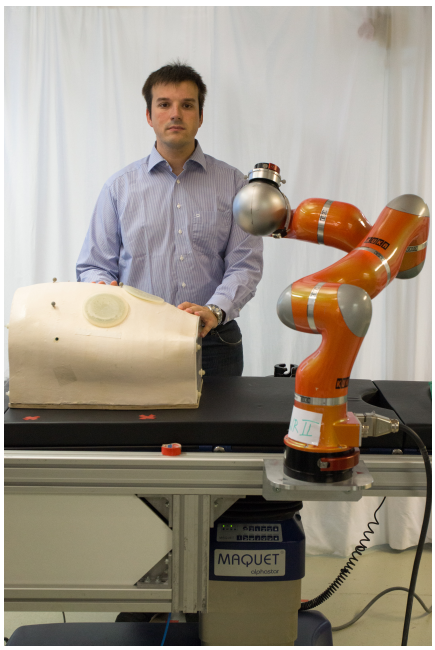


Figure 4.4.: Frontal view of human with occluded lower body parts

occur temporarily. In chapter 5.1.4 it is described, which methods have been applied to cope with the interferences, based on the requirements for the tracking and detection system.

The system architecture is based on unpublished work of Nicolai. For the Kinect™ 360 system two slave computers have been used, where each is driving two connected Kinect™ 360 cameras. A dedicated master, which is connected to the slave PCs using gigabit ethernet, is used for the point cloud generation, data fusion and object/people detection as well as tracking. The used approach theoretically allows for an unlimited number of cameras, compared to an approach where all cameras are connected to a single

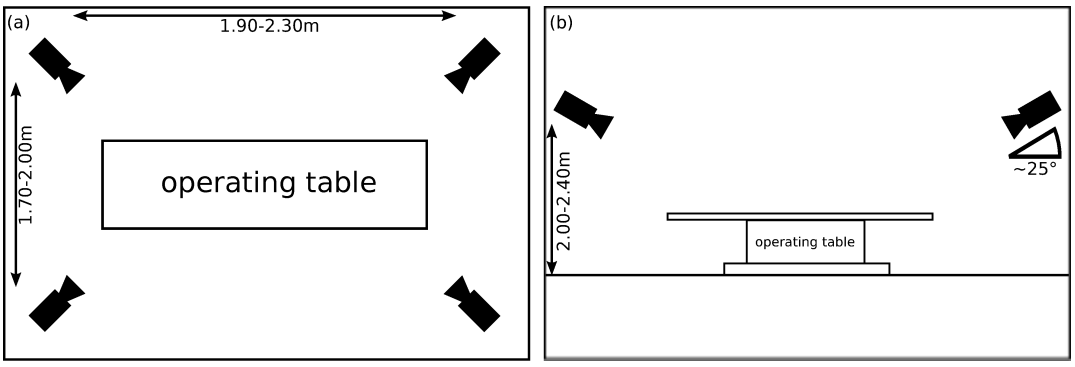


Figure 4.5.: (a) top view of the camera configuration, (b) side view of the camera configuration

computer, by using a distributed system that uses ROS over ethernet for communication.

Additionally, the system is scalable and the cameras can be placed farther away from the computer in charge of computation, compared to a single computer approach, where all cameras have to be connected to the processing computer and where cable lengths are limited by the maximum length of USB connections as used by the Kinect™ cameras. The Kinect™ 360 system is based on standard gigabit ethernet, which limited the number of cameras to four, whilst already introducing a notable delay of >900 ms as can be seen from the work of Nicolai et al. [95]. The data transmission rate of the full point cloud of a Kinect™ 360 camera is 40.32 Megabyte per second and can be computed using the 640*480 pixels resolution of the 11 bit depth map, the 640*480 pixels resolution of the three channel 8 bit color camera and the framerate of 30 Hz of the Kinect™ 360.

For a four camera configuration this sums up to an amount of 161.28 Megabyte per second, which is already above the 125 Megabyte per second maximum transmission rate of gigabit ethernet, limiting the theoretical update rate for the entire system to 23.25 Hz

The Kinect™ 360 system is based on the OpenNI and the NITE (Primesense, Tel-Aviv, Israel) frameworks, which have been acquired by Apple (Cupertino, CA, USA) in 2013. The Kinect™ 360 system runs on Ubuntu Linux (Canonical, London, UK). The people detection algorithm of NITE uses a model based approach that is very sensitive to movements, but which performs poorly when people do not move or only small movements are present. The slave computers of the Kinect™ 360 system are only used to collect depth and RGB images from the connected Kinect™ 360 sensors. The complete processing of the data is handled on the master machine.

The redesign of the system using the Microsoft Kinect™ One cameras lead to improved results compared to the Kinect™ 360 system. Due to the superior human detection and tracking quality of the Microsoft Kinect™ SDK, which uses an RDF-based approach to detect and track people from single depth images, the Kinect™ SDK replaced the OpenNI and the NITE framework in this work. Every Kinect™ One utilizes its own slave PC, which is built as an entirely passive solution to avoid disturbances of the airflow in the operating room by fans. The Kinect™ One though is equipped with a fan. Experiments in a closed housing revealed that the Kinect™ One can be sealed from its surrounding without overheating. This design change is crucial to allow for future experiments in real operating rooms and is the foundation of the feasibility of the approach for the operating room. All computers have been equipped with 10 gigabit ethernet adapters to avoid the bottleneck of a surcharged ethernet connection. Additionally, significant parts of the processing chain have been outsourced from the master to the slave

computers, which in the Kinect™ One version have enough performance to compute the full body tracking and to compute the colored point cloud as well as extracted point clouds that represent the detected humans. This reduces the amount of computational power needed by the master computer.

The computed data rate for the complete 3D point cloud of a Kinect™ One is 97.6896 Megabyte per second and can be computed using the depth resolution of 512*424 pixels of the Kinect™ One, where every point cloud is composed of 32 bit floating point values for x,y and z coordinates as well as 8 bit color information for the red, green and blue channels. The framerate is 30Hz.

With a four camera configuration this results in a data rate of 390.76 Megabyte per second which is well below the maximum data rate of 1250 Megabyte per second for 10 gigabit ethernet. This allows to easily transmit all the skeletal information together with the entire point cloud and the extracted point clouds representing the detected humans.

In order to allow for the registration of the cameras to each other, two algorithms have been designed, whereas one is able to allow for a pairwise registration of 3D cameras with respect to each other in order to be independent from an external referencing system. The other algorithm relies on the tracking information of an external tracking system and offers the advantage to register to a globally used reference system. Both algorithms are described in chapter 5.1.1 and results are given in chapter 6.1.2.

Compared to the literature shown in chapter 2, both systems are able to fuse point clouds of corresponding users from multiple views and multiple full body tracking information for at least six humans in the field of view, based on the positions of the point clouds that represent the detected users. Other systems are either able to fuse the point clouds of a single user in the field of view, or are only able to perform fusion when reliable tracking data is available from all cameras. The presented system however can fallback to a point cloud fusion when no tracking data is available and can additionally fuse the tracking information as soon as such information becomes available. This adds an increased detection performance, as the complexity and uncertainty of the detection of a human is lower compared to reliable full body tracking, resulting in a higher availability of point clouds representing the humans compared to the human's skeletal configuration. Both camera systems rely on the same imaging pipeline, which is presented in 5.1.2, which is applied to filtered point clouds from the camera systems. The filter algorithms which are presented in chapter 5.1.4 are slightly different for both systems, as the noise level of the Kinect™ 360 is higher compared to the noise level of the Kinect™ One, which in contrast to the Kinect™ 360 has to cope with flying pixels [109], due to the underlying ToF working principle.

The fusion algorithm described in chapter 5.1.6 allows to fuse full body tracking information of a human detected by multiple cameras in order to improve the tracking quality and reliability.

A virtual reality scene representing the observed operating theatre scene is used for the generation of features, for the detection of actions and as sensing component for the workflow-based controller. It can also be used to perform path planning, for a risk assessment of the current situation, or for the optimization of robot control strategies. For this purpose the fused point clouds from the perception system can be loaded into the virtual scene together with known objects in the same coordinate system, such as robots, tracked by an optical tracking system. The known objects in this scope can either be represented by meshes or by point clouds. A GPU based algorithm described in chapter 5.1.7 is then used to compute distances between point clouds and/or meshes. The distances from this algorithm form the basis for the situation detection algorithms in order to react to present or upcoming situations.

The distances and the tracking information can be produced by both camera systems. However the Kinect™ One based system is superior to the Kinect™ 360 system, as can be seen in chapter 6. Both systems can be used simultaneously to introduce failure safety through redundancy and to enhance the detection and tracking quality of humans present in the scene. The field of view of the square-shaped camera configuration has proven to cover all relevant areas in the surrounding of the operating room table.

Using the output of the camera system as input for probabilistic or rule based situation detection modules to interpret the data, the workflow-based controller can be provided with relevant information about prevailing situations, in order keep track of track the workflow. In this work e.g. the method of Schreiter [112] has been used for the detection of the intention of the surgeon to use the robot in hands-on mode. Other methods such as the methods shown in [36] and [49], can potentially be implemented and provided with the camera data to enlarge the set of detectable situations for medicine and other applications.

As the camera information solely is not enough to detect all potential and relevant situations for the workflow tracking, the camera system is supported by the use of additional sensors such as the robots' included sensors or the sensors included in the haptic input devices. For small objects or objects, whose coordinate frame has to be found with a high accuracy, a marker-based tracking system is used.

Some actions of the surgical staff may still not be detectable by the present sensors or components, as e.g. instrument tracking was not scope of this work. The systems to detect such actions are in this work supplemented by manual

interaction of the surgical staff with the system using a GUI, which can also be used as a fallback solution in case of a failure of one of the components.

4.4. Target workflow

Bimanual laparoscopic telemanipulation has been chosen as the target scenario for this work as it is of highest clinical relevance such as shown in [137], [14], [61], [46] and [65]. However, current telemanipulation systems are not able to include information about prevailing situations and most of them are not able to utilize information about the current state of the procedure.

The target workflow is composed of an automatic calibration of the robotic subsystems, automatic prepositioning, human-guided hands-on position, conventional telemanipulation task and finally an autonomous action to drive the robots to predefined finishing position. The workflow is executed on the workflow-based controller, which allows for the automatic selection of suitable components for the task execution and the automated tracking of the workflow without intentional interaction of the surgical personnel with the workflow-based controller. This results in an natural interaction with the system, which is able to perform according to the situation and the surgical personnel's needs.

The contents of the workflow modelled are presented in the following:

- **Robot calibration:** The robotic system performs a self calibration in order to localize the robot positions in the operating room. This task is performed for two robots as they are used in the bimanual telemanipulation scenario.
- **Definition of the Trocar positions:** After both robots are calibrated, the surgical personnel can then define the positions of the trocars for the insertion of the surgical instruments.
- **Automatic prepositioning of the robots:** The robots perform an automatic prepositioning task to allow the surgical personnel to easily align them with the operating field.
- **Hands-on insertion of the robots:** The surgical instruments attached to the robots are inserted by the personnel using the hands-on control mode of the robots. These tasks can be repeated until desired positions for the instruments are reached.
- **Telemanipulation:** The actual telemanipulation task is performed by the surgeon at a surgeon console. The surgeon console is equipped with a monitor, to show the image from the endoscopic camera, and two

input devices, which allow to control the robots in a similar manner than the da Vinci™ robot does.

- **Hands-on refinement of the robot position during telemanipulation:** The surgical personnel can refine the position of the robotic manipulators during the procedure to assist the operating surgeon.
- **Hands-on extraction of the robots:** The surgical instruments attached to the robots are extracted from the body of the patient using the hands-on control mode of the robot .
- **Autonomous robot movement to preplanned finish poses:** The robots move to preplanned finish poses in order to enlarge the space around the patient and to allow to remove the robots at the end of the procedure

At design time of the workflow no components of the actual execution system are linked to the tasks of the workflow itself. All requirements of the tasks are identified during run-time, which facilitates that the surgical personnel can freely define the workflow with little knowledge about the actual system. However, this underlines the importance of a consistent and complete knowledge base. A workflow as the one designed can only be executed when the tasks and transitions between the tasks are modelled in the knowledge base and when there are components available that are able to execute the actual tasks. In terms of the implemented workflow, the system is e.g required to find suitable calibration systems, robots, cameras, input devices and algorithms. Additionally, it has to deal with dependencies. The system is designed without an abstraction between the medical and the technical tasks. Therefore in the current implementation technical and medical tasks are mixed within the workflow. The implemented atomic tasks are described in chapter 5.2.2 and are included to the implementation of the workflow for the YAWL engine, shown in chapter 5.2, and in the knowledge base for execution on the workflow engine using the workflow-based controller.

5. Implementation

The following chapter is intended to describe the internals of the proposed approach, the used algorithms, their actual implementation and the integration with the system. It is shown how the workflow-based controller processes the knowledge from the knowledge base and the knowledge base is described in detail. The internal structure of the perception system as well as the calibration and registration methods are shown. The same applies for the realization of the workflow-based controller.

5.1. Perception system

As shown in chapter 4, the perception system is designed on top of an ethernet based distributed system. The software system is modularized, based on the actual requirements for the perception tasks and is divided into the following component categories:

- **Drivers:** serve as drivers for the connected cameras and provides the images to the ROS based system.
- **Point cloud processors:** use depth images from 3D sensors to compute point cloud representations of the observed scene.
- **Filtering algorithms:** allow to filter point clouds or depth images to reduce the noise in the data.
- **Human detectors and trackers:** use the cameras' output to compute point cloud or depth image-based representations of the humans observed by the cameras. Additionally, the components allow to extract 6 DoF full body tracking information of the observed humans.
- **Registration components:** allows to extrinsically calibrate multiple sensors to allow the representation of the information in a common coordinate frame.
- **Fusion algorithms:** Combines the information from the nodes described above, to get a global representation about human positions and skeletal configurations.

- **Feature extraction algorithms:** use the fused data from the fusion algorithms and known objects from the scene to extract features representing the system's state in real time.

For the Kinect™ 360 system the OpenNI framework has been used within a ROS node to allow for the capturing of the camera data. For the Kinect™ One system, the Microsoft SDK has been used within a ROS node to allow for the capturing of the camera data. The methods and algorithms used for the camera systems are described in the following chapters.

5.1.1. Registration

As written in chapter 4.3 two registration algorithms have been designed in order to allow for the registration with an external tracking system but also to allow for a pairwise registration between sensors without the need of external referencing hardware. The algorithms are described in the following:

5.1.1.1. Pairwise registration of cameras

The pairwise registration algorithm is designed to allow for the registration of RGB-D cameras without the need of an external referencing system. The algorithm relies on the RGB camera data, as well as on the depth camera data of RGB-D cameras and utilizes a checkerboard as calibration object. In order to allow for the pairwise registration one of the RGB-D camera's base frame has to be defined as the reference frame $K_{ref} = K_1$. In the target scenario, with four cameras the remaining cameras' coordinate frames are called K_2, K_3 and K_4. This results in three pairwise registration operations for the camera pairs (K_{ref}, K_2), (K_{ref}, K_3) and (K_{ref}, K_4). In order to allow for the registration process of the Kinect™ 360 cameras, as implementation of a RGB-D camera, the intrinsic and extrinsic factory calibrations by Microsoft have been used. Using the assumption of a perfect intrinsic calibration of both, the depth and RGB cameras as well as a perfect extrinsic calibration of the depth to the RGB camera, every pixel in the RGB camera corresponds with a depth value from the depth camera. This relation is used to detect the calibration object within the RGB camera frame and afterwards to get the 3D positions of known parts of the calibration object to allow for a calibration between the cameras in a camera pair. A checkerboard that is not point symmetric is used to allow for the unambiguous detection of the checkerboard orientation. The approach has been evaluated using the Kinect™ 360 system. The results are shown in chapter 6.1.2. The calibration process is described in the following:

1. Placement of a checkerboard in the field of view of both cameras of the camera pair to be registered. An image of the checkerboard is captured using the RGB sensors of both RGB-D cameras. OpenCV [27] is used to detect the corners of the checkerboard using the *findCheckerboardCorners* algorithm. The location of all corners of the checkerboard in x and y direction are extracted from the images.

2. For every pixel representing a corner of a checkerboard, the corresponding pixel from the depth camera is extracted using the intrinsic and extrinsic calibration between the IR/depth and the RGB sensor.

3. Using the known geometry of the lens of the IR sensors and the characteristics of the sensor, a 3D location for every checkerboard corner can be computed in the depth frames of the RGB-D cameras.

4. As the Kinect™ 360's depth data is very noisy, steps 2 and 3 are repeated for 60 depth frames of both cameras and the 3D locations of the checkerboard corners are averaged over the 60 frames. The locations of the corresponding corners of the checkerboard in both depth camera frames are stored as correspondences.

5. The checkerboard is relocated and steps 1-4 are repeated in order to acquire a larger set of correspondences distributed over the complete overlapping field of view of the cameras in the camera pair. This is performed to get a good registration result in the complete overlapping field-of-views (FoV).

6. After the collection of checkerboard corners at 5-13 checkerboard positions, the correspondences are used to estimate a transformation between the cameras. For this purpose a covariance list is incrementally built. The rotational component is then estimated using eigenvalue decomposition of the covariance, whilst the translational component is estimated using the means of the point sets. This process can be compared to the estimation process for the transformation within the ICP algorithm [20]. In this work the TransformationFromCorrespondences implementation of PCL has been used.

If repeated for all four pairs of cameras, all point clouds acquired by the camera systems can be represented in the depth frame of the reference camera to allow for a global representation of the 3D scene. As the Kinect™ 360 cameras cannot be synchronized to each other and as the corner locations are averaged over 60 frames, it is required that the checkerboard is in a stable position during the capturing of both, the RGB and the depth images.

5.1.1.2. Registration with respect to an external optical tracking system

In contrast to the pairwise registration of cameras the algorithm described in the following requires an external referencing system. For this purpose a referencing body with known geometry and a defined coordinate frame as well as mechanical, optical or electromagnetic tracking systems can be used. In the scope of this work the ARTtrack2 system (Advanced Realtime Tracking GmbH, Weilheim i. OB, Germany) has been used, which allows to detect up to 20 rigid bodies consisting of at least 4 retroreflective marker spheres, which have to be in a unique configuration for each of the 20 rigid bodies. The algorithm allows to register multiple cameras even in case the field of view of the cameras does not overlap. An overlap with the working range of the external referencing system with the FoV of the camera to be registered is required. Additionally, the approach is able to work with 2D cameras as no depth information is required.

The approach has been evaluated with the Kinect™ 360 as well as with the Kinect™ One system and the results are shown in chapter 6.1.2. As the algorithm relies on 2D data only, a precise object shape and size detection within the 2D camera frame is required to allow for a good registration accuracy. Therefor an intrinsic calibration of the cameras to be registered has been performed using the ROS package *camera_calibration* [4]. As the ART tracking system used is only able to retroreflective markers and a tool is required to touch the checkerboard corners to allow for the capturing of the corners position within the base frame of the ART tracking system, an NDI pointer as shown in Fig. 5.1 has been used for the calibration process.

Figure 5.1.: The NDI pointer used in this work

To allow for the computation of the tip pose of the pointer the pointer's tip position has to be calibrated. This is performed using pivoting of the pointer's

tool tip in the working space of the tracking system, which means that the tool tip is fixed while the coordinate frame given by the optical markers is moved around the tip on a sphere shaped surface as can be seen in Fig. 5.2. The position of the pointer's coordinate frame is continuously captured

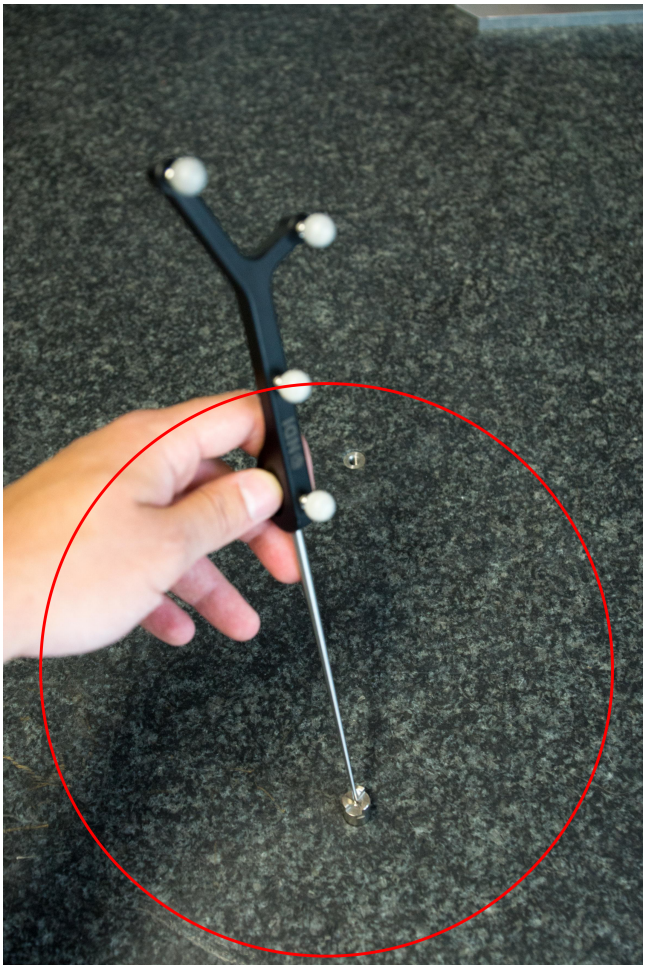


Figure 5.2.: The pivot motion in red around the fixed pointer tip

during this process and the radius of the sphere as well as the position of the center of the sphere with respect to the pointer's coordinate frame are then computed using the least-squares-minimization method. The pointer tip is located in the center of the sphere and the transformation from the body to the tip is therefor known from the minimization. After the intrinsic camera calibration and the pointer's tip calibration the camera registration process is performed as follows:

1. The checkerboard is placed within the field of view of the camera to be registered and the working range of the tracking system.
2. An image of the checkerboard is captured using the IR camera and the corner locations of the checkerboard are computed using OpenCV's

findCheckerboardCorners algorithm. The position of the checkerboard corners are then refined using OpenCV cornerSubPix algorithm.

3. The OpenCV solvePNP algorithm, based on the Levenberg-Marquardt optimization [82], is used to find the pose of the known 3D geometry of the checkerboard within the captured set of 2D coordinates representing the checkerboard positions. For the 3D representation of the flat checkerboard 0.0 m is chosen as z coordinate of the point set in the found checkerboard coordinate frame. The pose of the checkerboard is averaged over 60 frames to get a more stable pose.

4. The four outer corners as can be seen in Fig. 5.3 are touched with the pointer's tool tip and the locations are stored for correspondence estimation.

5. From step 3 the 3D checkerboard pose with respect to the camera frame is known. The known checkerboard geometry and the pose is used to compute the pose of the four outer corners in the camera frame. The poses are associated to their corresponding values from step 4.

6. The checkerboard is relocated and steps 1-5 are repeated in order to acquire a larger set of correspondences distributed over the complete overlapping field of view of the cameras and the working space of the tracking system.

7. After the collection of checkerboard corners at 5-13 checkerboard positions step 6 of the pairwise camera registration algorithm is applied, which computes the transformation between the base frame of the tracking system and the optical frame of the IR camera.

8. The infrared camera's coordinate frame is identical to the depth frame of the Kinect™ cameras. This means that the computed transformation from the prior step is the transformation between the depth frame and the reference frame of the optical tracking system.

Similarly to the pairwise registration algorithm, the process has to be repeated for all cameras in order to be able to construct a point cloud representation of the actual scene within the frame of the optical tracking system, which is the reference frame. If all cameras are registered with respect to the tracking system, both Kinect™ camera systems can be used within the optical tracking system's coordinate frame, allowing to combine features extracted from point clouds with trajectories of marker-equipped objects. This data can then be used to analyze movements in the scene or to compute the geometrical relations between the objects being perceived. This algorithm has been used for the final system evaluation.

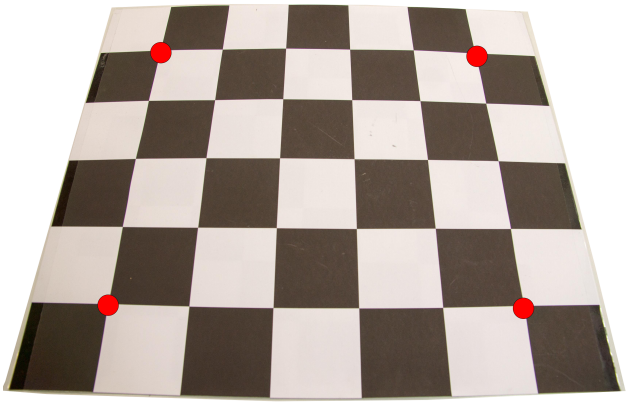


Figure 5.3.: A typical checkerboard as used for registration. The outer corners are marked with red circles.

5.1.1.3. Calibration of the floor plane

The point cloud fusion algorithm described in chapter 5.1.5 requires information about the location of the ground plane in the reference coordinate system. The floor plane therefor has to be detected, which is performed using the PCL implementation of the Random Sample Consensus Algorithm (RANSAC) [41] and the PCL plane model as well as the PCL implementation, which delivers the coefficients of the plane in the Hesse normal form. The algorithm performs the segmentation of the most dominant plane, which fits the used model best. As the most dominant plane in some scenarios may not be the floor plane, such as in presence of an operating table, the calibration is performed without the presence of the operating table. However, with the known height of the cameras above the floor, the parts of the point cloud the plane is searched in, can be restricted using a pass through filter. This filter removes the points of a point cloud that do not fit into a range given to the pass trough filter and can be used to assure that the correct plane is being found even in presence of other planes. The parameters of the floor plane are then stored in the Hesse normal form for usage in the fusion algorithm.

5.1.1.4. Registration of the robot

In order to allow for the detection of the position of the robots in the physical scene, the marker-based optical tracking system is used. Therefor the flange of every robot is equipped with a rigid body for the optical tracking system as can be seen in Fig. 5.4.

The transformation from the flange of a robot to the attached rigid body has to be found. For this purpose, a two step calibration process has been developed.

Figure 5.4.: The flange of a lightweight robot with markers on a rigid body.

In step one the position of the flange of the robot with respect to the rigid body is computed using the same algorithm that is used for the pointer's tip calibration as shown in chapter 5.1.1.2.

This is performed by placing the robot's flange with the attached rigid body in the field of view of the tracking system. The orientation of the flange is randomly changed while the position of the flange is kept constant using the inverse kinematics of the robot. This causes the coordinate frame of the rigid body, attached to the robot, to move on a sphere around the position of the flange. This is repeated 75 times and the position of the coordinate frame of the rigid body is stored for all of the 75 poses. Using the least-squares minimization method the position of the flange of the robot with respect to the coordinate frame of the rigid body, which is constant as the body is rigidly attached to the robot's flange, is found. In order to estimate the orientation of the flange, a second step has to be performed. Using the kinematics of the robot, the position of the flange of the robot with respect to the base frame of the robot is known. Additionally the position of the robot's flange in the coordinate frame of the optical tracking system is known by using the pose of the rigid body $K_{ots,body}$, attached to the robot, in the reference frame of the optical tracking system and the vector from the coordinate frame of the rigid body to the flange of the robot, known from the least-squares estimation. In order to collect corresponding points the robot is moved to 75 random positions with a fixed orientation in a cube shaped workspace of dimensions 0.1 m x 0.1 m x 0.1 m while optical tracking is active. The position of the flange in the coordinate frame of the optical tracking system and in the coordinate frame of the robot's base are stored as correspondences. The TransformationFromCorrespondences algorithm from PCL, which is also used for the registration of the camera systems, is then utilized to compute the transformation from the base of the robot to the optical tracking system's reference frame. This transformation is

called $K_{base,ots}$, while the transformation from the robot's base to its flange is called $K_{base,flange}$ The transformation from the rigid body to the robot's flange $K_{body,flange}$ is then computed using equation 5.1. The transformations used are shown in Fig. 5.5.

$$K_{body,flange} = K_{ots,body}^{-1} * K_{base,ots}^{-1} * K_{base,flange} \tag{5.1}$$

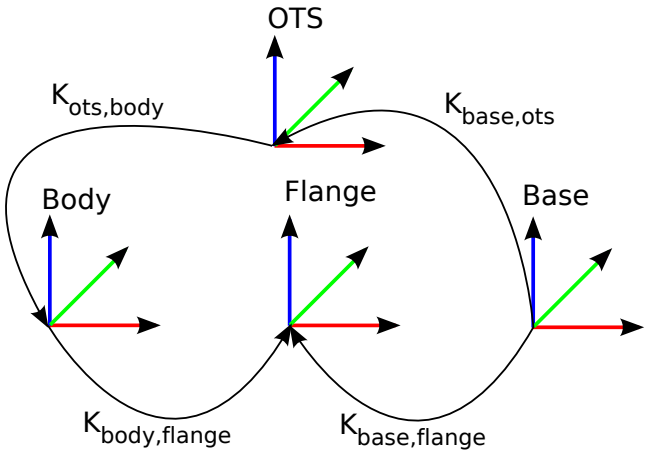


Figure 5.5.: The transformation chain for the robot registration algorithm.

5.1.2. Imaging Pipeline

The imaging pipeline of the camera system shown in Fig. 5.6 is utilized to compute precise point cloud and skeletal configurations of humans and finally to compute distances between objects and humans that are situated in the field of view of the camera system. The detection, tracking and filtering steps are computed with 30 Hz and are triggered for every frame acquired by each of the cameras. All steps after the filtering, including the fusion process, are computed with 15-25 Hz, depending on the number of objects and humans perceived by the camera system. All steps of the imaging pipeline are described in the following paragraphs.

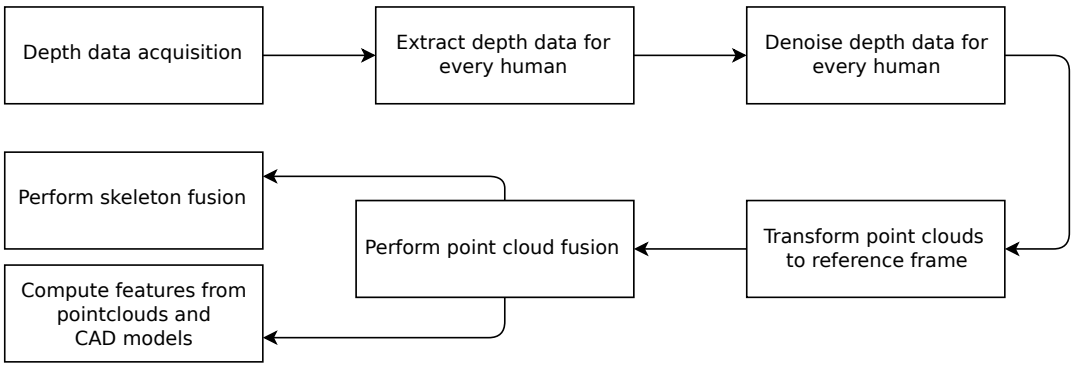


Figure 5.6.: The imaging pipeline of the perception system.

5.1.3. Detection and Tracking

The Kinect™ 360 and the Kinect™ One system differ in the human detection and tracking method used. The Kinect™ 360 system uses the model based NITE approach by Primesense (Tel-Aviv, Israel), which is highly dependent on the movement of the humans in the scene. Additionally, the NITE approach is a two step algorithm. In step one, the human is detected and the pixels corresponding with a human are labelled. In step two, the algorithm utilizes the labelled depth images to fit a human skeleton into the labelled pixels that represent the human. Only if step one is successful one can compute a 3D point cloud representation of the detected human. If step two is also successful, the 6 Dof pose of 15 skeletal joints of the human can be computed. The NITE algorithm requires a sequence of images both for detection of humans and for computing the skeletal configuration of the human. As the skeletal tracking is based on the detection step, the point cloud-based representation is available earlier compared to the skeletal information after a human entered the scene. Additionally, the point cloud-based representation is more reliable. NITE can detect up to 16 humans at once and labels each pixel in the depth image with the id of the human to which it corresponds. In the Kinect™ 360 system, the labelled depth image is then divided into n depth images whereas n is the number of humans detected by the camera system. Each of the resulting depth images contains all depth pixels representing a human. All other depth values are set to 0. The result can be seen in Fig. 5.7. The extracted depth images are then filtered using a morphological erosion filter with rectangular mask of size 5x5 pixels. This removes most of the noisy pixels at all borders of the human represented in the depth image. As all other parts of the image, except the parts representing the human, are zero, the operation does only affect the border of human within the image. After the filtering process the known parameters of the depth camera of the Kinect™ 360 and its lens, as well as the relation between depth values and meters are used to compute point clouds from the extracted depth images. The

Figure 5.7.: The extracted depth image for a single human in a single depth frame from a Kinect™ 360 camera.

resulting point clouds are then filtered using the statistical outlier removal filter described in chapter 5.1.4 and each of them represents a detected human as can be seen in Fig. 5.8.

In contrast to the Kinect™ 360 system, for the Kinect™ One system ,the Microsoft algorithm built in to the Kinect™ SDK is used. This approach is RDF-based [114] and extracts both, the pixels representing the user and the skeleton from a single depth image. As the noise characteristics of the Kinect™ 360 and the Kinect™ One differ as described in 5.1.4, the noise removal process for the depth images is not required and point clouds, representing every human detected, are directly computed using the Microsoft SDK without the need of depth images. The Microsoft framework allows for a more robust and faster people detection compared to the NITE based approach. However, it allows to track only up to 6 people within the field of view of a single camera. The Microsoft skeleton tracking approach delivers a skeletal model including up to 21 joints of a human.

The outcome of the detection and tracking process are similar for both camera systems. Both deliver a point cloud for each detected user in the field of view of the camera the algorithm is applied to, as well as skeletal information for the humans being tracked. The quality of the Kinect™ One system's output is superior the one of the Kinect™ 360 system.

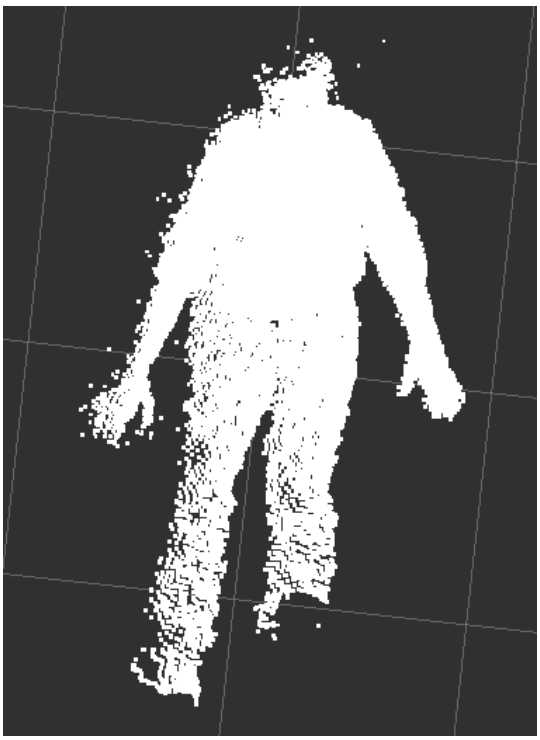


Figure 5.8.: A point cloud representing a human as seen from a Kinect™ 360 camera.

5.1.4. Filtering

Both, the Kinect™ 360 as well as the Kinect™ One camera system do not provide point clouds that are free of noise and the noise characteristics of the Kinect™ 360 and the Kinect™ One differ. However, the point cloud noise filtering algorithm for both systems is similar as the most relevant part of the noise occurs at object boundaries in the point clouds. The noise characteristics of the Kinect™ 360 is described through pixels at edges that lie between foreground and background in unoccupied space. In contrast to this, the Kinect™ One suffers from the flying pixels phenomenon that is common to ToF systems [109], which is described by pixels at edges in the depth image that change their location from foreground to background and vice versa. The raw colored point cloud from both systems is depicted in Fig. 5.9. Additionally, the noise level of the Kinect™ 360 significantly increases when more than one camera is used as can bee seen in [40]. This problem is tackled by the pre-filtering described in chapter 5.1.3.

As actual filtering algorithm, a statistical outlier removal filter is used, which is designed to remove sparse outliers of points in a point cloud. For this purpose, the algorithm performs a statistical analysis of the neighborhood of all pixels within a point cloud. Points that do not match the defined neigh-

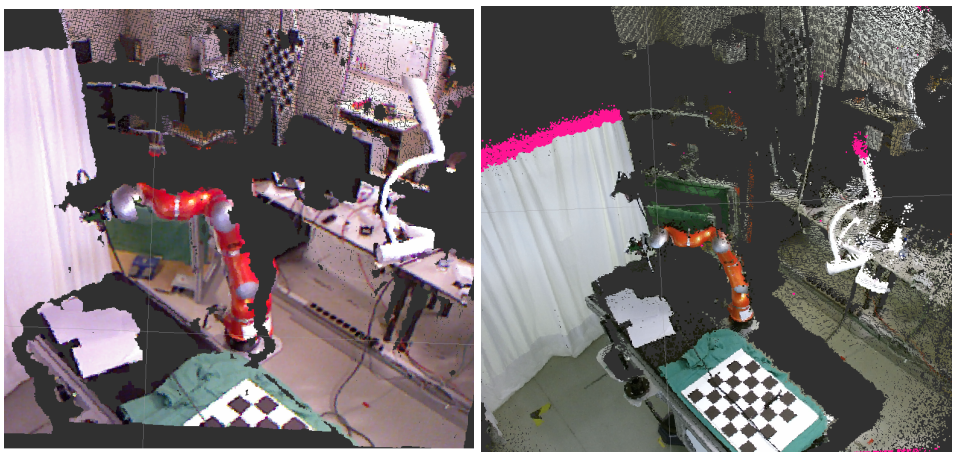


Figure 5.9.: Left: raw colored point cloud from a Kinect™ 360 camera. Right: raw colored point cloud from a Kinect™ One camera.

borhood criteria are then removed. The algorithm used is implemented in the PCL and is called StatisticalOutlierRemoval. In a first step, the algorithm computes the mean distances of every point to its neighbors using a k-d tree [19]. It is then assumed that the distribution of the mean distances to the point's neighbors is gaussian. The algorithm then computes the mean values over the mean distances of all points of the point cloud and removes all points, whose mean distance to their neighbors is outside a defined range. This range is defined by a standard deviation multiplier. The threshold Tr for points to be considered outliers is then computed using equation 5.2.

$$Tr = \mu + \alpha * \sigma \tag{5.2}$$

in which α is the standard deviation multiplier, μ is the mean and σ is the standard deviation. All points with a mean distance to their neighbors that is bigger then the computed threshold are then considered as outliers and are removed from the point cloud.

In order to achieve a nearly noise free point cloud for the Kinect™ 360 system a standard deviation multiplier of 1.0 has been used and 50 neighbors have been considered for the computation of the distribution. For the Kinect™ One, a standard deviation multiplier of 0.7 has been used and 60 neighbors have been considered for the computation of the distribution.

5.1.5. Point cloud fusion

The outcome of the detection and filtering process is a set of point clouds representing all detected humans in all four camera frames. The total set of

point clouds is called N, which is computed using equation 5.3.

$$N = N_1 \cup N_2 \cup N_3 \cup N_4 \tag{5.3}$$

where N_1 to N_4 are the point cloud representations of users detected by camera 1 to camera 4. If humans are situated in overlapping fields of view of 2 or more cameras, multiple point clouds in N are representing one and the same human. The algorithm described in the following identifies point clouds that correspond to the physical humans and performs a fusion step in order to compute a point cloud representation including multiple viewpoints of a human. The final set of point clouds equals the number of physical users in the scene that are observed by at least one of the cameras. The algorithm is based on the computation of centroids of point clouds. The steps of this fusion algorithm are shown in the following:

1. The transformations computed from the registration step are used to transform all point clouds representing humans into the reference coordinate frame.
2. For each point cloud the centroid is computed by summing up the location vectors of all points in the point cloud and by dividing it by the number of points in the point cloud as can be seen in equation 5.4 where p_i is a point and m is the number of points.

$$c = \begin{bmatrix} \frac{p_{1,x}+p_{2,x}+...+p_{m,x}}{m} \\ \frac{p_{1,y}+p_{2,y}+...+p_{m,y}}{m} \\ \frac{p_{1,z}+p_{2,z}+...+p_{m,z}}{m} \end{bmatrix} \tag{5.4}$$

   This results in a centroid c for every point cloud $n \in N$. The set of centroids is then called C whereas $C = C_1 \cup C_2 \cup C_3 \cup C_4$ and C_1 to C_4 are the sets of point clouds computed from N_1 to N_4.
3. Then all possible combinations of pairs of centroid sets in C are built: (C_1, C_2), (C_1, C_3), (C_1, C_4), (C_2, C_3), (C_2, C_4), (C_3, C_4) and the euclidean distances between all possible combinations of centroids in the pairs are computed.
4. Thresholding is applied to the computed euclidean distances. If an euclidean distance computed in a pair is below the defined threshold, the centroids of the point clouds for which the centroids have been computed are in proximity. For the target application as explained below this means that the point clouds represent one and the same human.
5. For all pairs of centroids that have been found to be in proximity, the associated point clouds are fused by concatenating their sets of points. The result is stored as a multiple view representation of a human detected by multiple cameras.

The simplification of the centroid as a representation for the center of the human can be made as the system uses the sagittal plane of the human for detection and skeleton tracking. In this view, a human that is standing still on both feet with hanging arms can be abstracted as a body, which is nearly symmetrical to the longitudinal axis from feet to head. In case of a perfect point cloud, representing the human, this means that the centroid will be placed along the longitudinal axis. This applies only for symmetrical poses such as the described one, which can be seen in Fig. 5.10. Additionally, when a frontal image of a human is captured in the operating room, the assumption can be made that there is little to no abduction of the legs of the personnel.

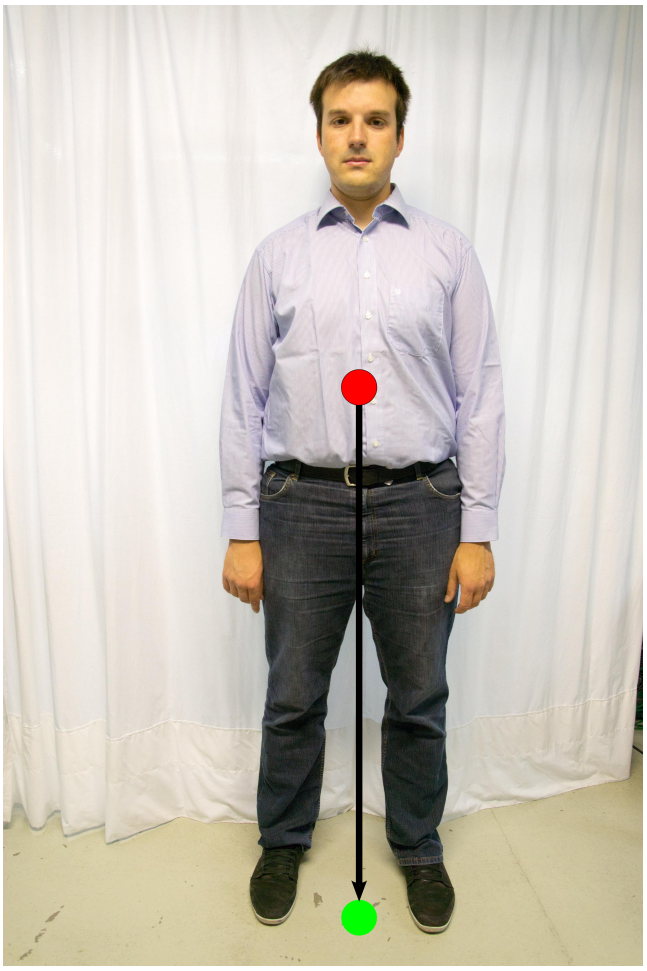


Figure 5.10.: An example for an symmetric pose with the probable centroid position (red) and the centroid position projected to the floor (green).

The head and the body usually are only slightly or not bent to the side. The only significant parts of the human body, which could lead to a movement of the centroid away from the longitudinal axis of the body are the arms of

the personnel or bending of the upper body. Compared to the rest of the body, the amount of pixels/points representing the arms of the personnel is relatively low and therefore arm movements affect the centroid positions only to a small extent. The legs of the humans cannot be seen in the surgical scenario as they are occluded by the operating table for the proposed camera configuration. In the optimal scenario, with a nearly symmetrical human posture the centroids of two or more point clouds representing the human will be at identical positions. In case of asymmetrical postures, where the centroid is slightly moved away from the longitudinal axis, thresholding can be used to find the centroids of point clouds of users that are close enough to be considered as the same physical human. All point clouds of which the centroids have a smaller distance than a defined threshold are therefore considered to belong to the same human.

In reality, the value for the thresholding has to be placed in the range of 0.4 m-0.6 m. These high values are due to the fact that for the given camera configuration, there cannot be more than one camera which is able to directly have a front view to the perceived human. Additionally, the data provided by the Kinect™ cameras is not 3D but 2.5D which means that the back of objects cannot be captured using a single camera system. Each camera can only perceive a part of the surface of a human operator and the centroid for the point cloud resulting from the depth image has its centroid placed close to the surface of the human. In the worst case, where two cameras capture the human while the human is situated in between them, one camera has a front view on the human and one camera has a back view. This is on one hand not optimal for the skeleton tracking and on the other it results in a centroid, which is situated at the humans back and another centroid, which is situated at the humans front. The effect of different placements of centroids is illustrated in Fig. 5.11.

The correct fusion is also hindered by partial views of a human operator. Such a partial view can be seen in Fig. 5.12. A partial representation of a human occurs when a camera's view on the human is occluded or when the human is partly out of the field of view of the camera. Depending on the missing parts of the human in a partial view representation, the position of the centroid relatively to the human is hardly predictable. This situation is likely to occur in crowded operating theatres as well as in the lab situation in which the camera systems have been evaluated.

Finally, additional complexity is introduced by the bending of the upper body of a human to the front, which is a common situation for the operating surgeon in the operating room. This causes the centroid to move to the front compared to the lower body parts. All described effects for the movement of the centroid away from the center of the human are not independent

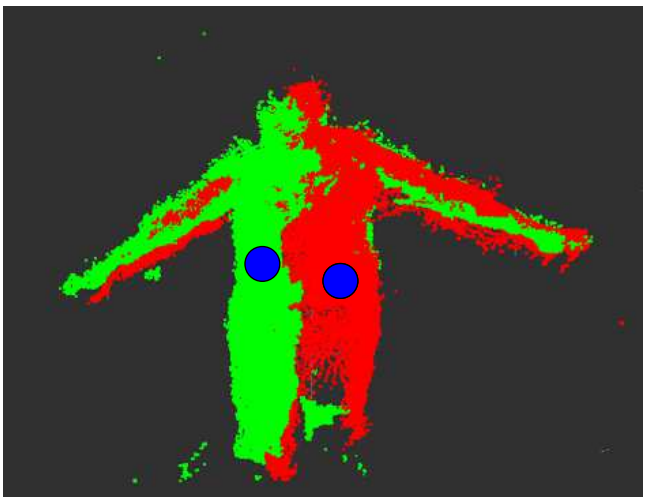


Figure 5.11.: Locations of centroids (blue) are depending on the viewing direction of the cameras. Two views (red and green) of one and the same user are shown.

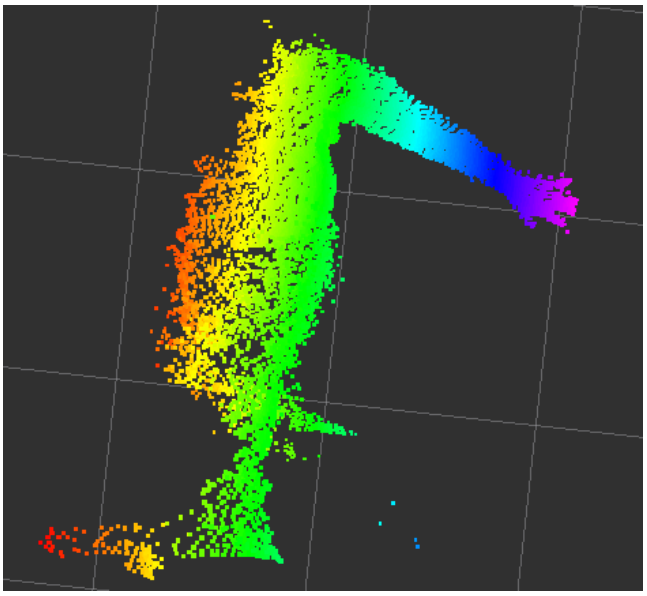


Figure 5.12.: Partial view of a human. Left side of the human is out of the field of view of the camera

problems, as the effects of viewing angle, bending and occlusions are affected by each other

The threshold has to be chosen carefully, as a threshold, which is too, low will cause false-negative results, resulting in falsely not fused point clouds, for the point cloud fusion and a threshold, which is chosen too high will increase the rate for false-positive results, resulting in wrongly fused clouds.

In order to keep the threshold as low as possible, the problem of the euclidean distance computation has been reduced to a two dimensional problem by using the fact that humans in the operating room can only move on the floor plane. The centroids are therefor projected to the floor plane and considered as the trace of the humans with reference to the floor (see Fig. 5.10). This process has been integrated to the algorithm before the thresholding. The Hesse Normal Form of the floor plane, known from chapter 5.1.1.3 is used to compute the normal. The computed centroids are then moved along the normal until they are in contact with the floor plane. This process removes the z-value (floor coordinates) and therefore one variable from the computation of the euclidean distance, which allowed to relax the threshold to 0.4 m and which resulted in good false-positive and false-negative rates.

The fused point clouds are then provided to the feature extractor. Finally, a correspondence matrix is built, which contains identifiers for all point clouds that have been identified to represent one and the same human. This information is also passed to the skeleton fusion to allow for the fusion of the skeletal information from multiple viewpoints.

5.1.6. Skeleton Fusion

Both camera systems are able to deliver full body 6 DoF tracking of the perceived humans. In case of the Kinect™ One system, the tracking information is usually available as soon as a human is detected. For the Kinect™ 360 system, the tracking information gets available after the NITE algorithm was able to fit a skeleton to the detected human. Similar to the presence of multiple point clouds from different points of views, which represent the same human, there can be multiple tracking information from multiple viewpoints. Every skeletal tracking set is associated to the point cloud of the related human. Using the information about corresponding users from the algorithm described in chapter 5.1.5 it is already known, which sets of tracking information belong to one and the same human.

Full body tracking information can sometimes be very noisy, e.g. in case the camera cannot directly see all body parts. Therefor the fusion of multiple tracking information can help to improve the tracking quality as it is likely that a body part which is occluded from one point of view is visible from another camera's point of view. Both the Kinect™ SDK's and the NITE SDK's tracking algorithms are designed for full body tracking from a frontal viewpoint on the sagittal plane of the human. In the target scenario it may be possible that one of the cameras has a view on the user's back, which likely causes a degraded tracking quality for this camera. In a square-shaped camera configuration as used in this work a maximum of two cameras can

have a view on the front of the human. Therefor the maximum number of tracking sets used for the computation of a multiple view skeletal tracking for a human is restricted to two sets. If only one set for a human is available, this tracking information will be considered as the global tracking representation for the human. If two sets are available, the tracking information of these sets is fused to compute the global tracking representation for the human. If more than two sets of tracking information are available, the two best fitting tracking sets are computed.

In this work, the most similar skeletal data have been regarded as the best fitting skeletons. For the computation of the most similar skeletal data, the lower body parts have been ignored as they are occluded by the operating table and tracking of these parts is therefor not reliable. For all remaining parts of the body and for every set of tracking information, the euclidean distance between the position of the body part and the position of the same body part in all other sets is computed. T has been defined to be the set of all available tracking information sets from all available viewpoints. B is the set of all body parts in all $t \in T$. Therefor, for every $t \in T$ there exists a $b_t \in B_t$, which represents a 6 DoF pose of a body part detected from one of the cameras. Hence t_1 includes all poses of all parts of the body $b_{1,1}$ to $b_{1,n}$ as included in tracking information 1 and t_m includes all poses of all parts of the body $b_{m,1}$ to $b_{m,n}$ whereas m is defined as the total amount of tracking information for a human perceived by the camera system and n is defined as the amount of body parts tracked. The euclidean distance in between the points p and q are defined as $d(p, q)$

The rotational component of the poses is neglected for the computation of the most similar tracking information. The euclidean distances between all parts of the body are therefor computed using equation 5.5

$$d(b_{1,1}, b_{2,1}), d(b_{1,1}, b_{3,1}), \ldots, d(b_{1,1}, b_{m,1}) \tag{5.5}$$

to equation 5.6.

$$d(b_{1,n}, b_{2,n}), d(b_{1,n}, b_{3,n}), \ldots, d(b_{1,n}, b_{m,n}). \tag{5.6}$$

This results in euclidean distances for all parts of the body in every combination of sets. For every part b_t of the body, it is checked which combination of sets offers the smallest euclidean distance. It is then counted, which combination of sets has the highest amount of smallest euclidean distances between all body parts in B_t.

An alternative to this is to sum up the euclidean distances between body parts for every combination of sets of tracking information and then to consider the sets of the combination with the smallest accumulated euclidean distances as most similar sets. This method however is prone to outliers and has therefore

been rejected. The combination of sets with the highest count of smallest euclidean distances between the body parts in the sets is then used to compute the global tracking representation of the human.

The computation of the global tracking representation of a human is computed individually for every body part in the sets of tracking information that have been chosen for fusion in the prior step. The translational component t_i for a global body part pose is computed using the average pose of the body part's pose in the sets by computing equation 5.7

$$\vec{t_i} = (\begin{pmatrix} p_x \\ p_y \\ p_z \end{pmatrix} + \begin{pmatrix} q_x \\ q_y \\ q_z \end{pmatrix}) \cdot 0.5 \tag{5.7}$$

in which p is the translational component of the body part in the first set and and q is the translational component of the second set. The rotational component r_i for a global body part is interpolated using quaternion spherical linear interpolation (SLERP) as can be seen in equation 5.8

$$r_i = (1 - t) \cdot r_1 + w \cdot r_2 \tag{5.8}$$

in which r_i is the interpolated quaternion, r_1 and r_2 are the rotational components of the body part in the sets of tracking information and w is the weight of r_2 which is set to 0.5 as both source quaternions are contributing equally to the result. This is repeated for every body part in the sets of tracking information and the result is provided as the global tracking representation of the human. The entire process is repeated for every human in the field of view of the camera system and is identically for both the Kinect™ 360 and the Kinect™ One camera system. As can be seen in chapter 6.1.2.4, the fusion process reduces the noise of the tracking information and increases the tracking reliability.

5.1.7. Feature extraction

As all point clouds are already in the same coordinate frame, they can be loaded to a virtual scene representation without further post-processing. The scene representation used in this work is capable of working with point clouds and CAD models. The viewer from PCL is used to visualize the current configuration of the scene. Measures can be extracted from this scene representation, which allow to judge about actions that take place in the physical scene. In this work closest distances between objects and humans have been used as primary information for the judgment about the scene. The most important distances in this scope are the closest distances between the elements of the robots in the scene and the body parts of the humans in

the scene. If the registration of the camera system is performed with respect to the optical tracking system, the coordinate frame used for the virtual scene representation is the base frame of the optical tracking system. In this case all marker-equipped objects for which CAD files exist and whose geometry with respect to the optical markers is known, can be added to the scene. This is performed for the KUKA lightweight robots for which CAD files exist and for which the position of the optical markers frame with reference to the robot is known from chapter 5.1.1.4. In order to estimate the robot's base position we therefor use the known transformation $K_{body,flange}$ from the marker-equipped rigid body to the robot flange and compute the transformation $K_{base,flange}$ from the robotics base to the flange using the Denavit-Hartenberg values and the measured joints of the robot. The base pose of the robot can the be computed using equation 5.9, whereas $K_{ots,body}$ is the transformation of the body as measured by the optical tracking system.

$$K_{ots,base} = (K_{base,flange} * K^{-1}_{body,flange} * K^{-1}_{ots,body})^{-1} \tag{5.9}$$

Using the known position of the robot's base, the Denavit-Hartenberg values, the measured joints of the robot and the assumption that there is only a small error from the kinematics, one can compute the CAD model for the actual robot configuration by transforming the points of each part of CAD model of the robot to their actual location in 3D space. This allows to visualize the robots position and configuration together with the point clouds measured by the camera system as can be seen in Fig. 5.13. The implemented methods allow to compute the closest distance between two point clouds, between two CAD models or between a point cloud and a CAD model. In case one or two, CAD models are used, the points are extracted from the CAD model and are treated as a point cloud.

As both, the NITE and the Kinect™ SDK algorithms sometimes consider the robot to be a human, point clouds are generated for the robots, which leads to a distance of 0.0 m between this point clouds and the falsely classified robot. To avoid this phenomenon, the shape cropping method as used in Nicolai et al. [95] is used, where first the CAD model of the robot is fattened to compensate for registration errors and second the points that are situated inside the CAD models are cut from the scene. Then the euclidean distance $d(p, q)$ for each $p \in P$ and each $q \in Q$, whereas P is the first point cloud and Q is the second point cloud, is computed, which results in $n \cdot m$ euclidean distances, whereas n is the size of P and m is the size of Q. The smallest of the computed euclidean distances is the closest distance between the two sets of points. In this work only high resolution point clouds and CAD models with high polygon count have been used, which guaranteed for a negligible error introduced due to the fact that no triangulation for the point clouds and no triangle information from the CAD models have been used. For high

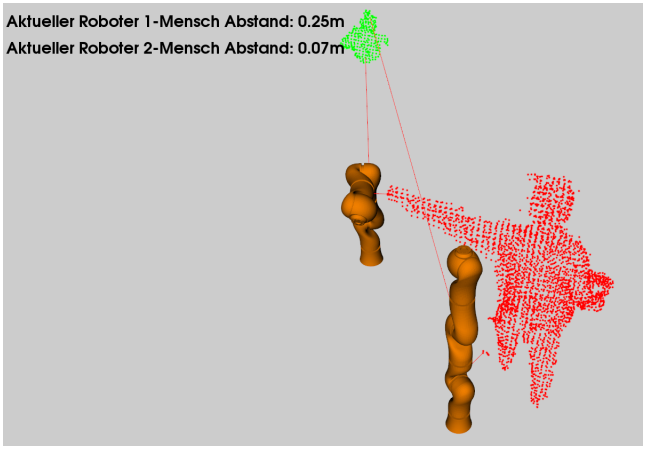


Figure 5.13.: 3D representation of the scene with the visualized robots and 2 humans, whereas one human is close to the robot (red) and one human is farther away (green). The distances of the human closest to the robots is shown in the in the left upper corner

resolution point clouds and CAD models, a huge amount of computations have to be performed, in order to find the points of the objects that are closest and to compute the euclidean distances between them. As the euclidean distances computation task can be computed in parallel, NVIDIA CUDA capable graphics adapters have been used for this purpose to speed up the computation compared to CPU-based euclidean distance computation. For this purpose the point clouds are loaded into the graphics memory of the graphics adapter and a CUDA kernel is used which computes the squared euclidean distances $d(p,q)^2$ for each $p \in P$ and each $q \in Q$ instead of the euclidean distances, which requires less processing power. Afterwards, the CPU is used to extract the points, which are closest to each other from the computed squared euclidean distances and to compute the square root from the result.

If the distance between a CAD model or a point cloud and a single point has to be found, the computation is entirely performed on the CPU as only as much computations, as points in the point cloud/CAD model have to be computed. This is e.g. the case if the distance between a robot and a body part from the skeletal information of a human is searched. In the most simple case, the distance between a point and a point is computed, such as the distance between a robot's flange and the hand of a human as extracted from the skeleton tracking.

For every frame collected by the camera system, the feature extraction algorithm computes:

- The closest distance between every human and every robot (closest

distance between point cloud representation of a human and a CAD model)

- The closest distance between every human and every robots' elbow (closest distance between a point cloud and a point which is computed using the robot's base position, Denavit-Hartenberg parameters and the measured joints)
- The closest distance between every human and every robots' flange (closest distance between a point cloud and a point, which is computed using the robot's base position, Denavit-Hartenberg parameters and the measured joints)
- The closest distance between every hand of the humans and every robots' flange (distance between a point from the skeletal configuration of a human and a point which is computed using the robot's base position, Denavit-Hartenberg parameters and the measured joints)
- For every robot: the closest distance between the human that is closest to the robot (closest distance between two point clouds).

If the history of these distances is considered, approaching speeds and accelerations can be computed using the first and second derivatives of the distances. This information is provided to the situation detectors.

5.1.8. Situation detectors

In order to detect prevailing situations, suitable detectors, which utilize the input information and a defined or learned rule set, have to be in place. Such detectors can be of various nature and can be implemented using a simple *if a then b* rule, more complex rules or by using probabilistic methods such as Hidden-Markov-Model (HMM)- or Random Decision Forest- (RDF) based classifications. In this work, simple *if a then b* rules have been used for easily detectable situations. For more complex situations, existent HMM-based solutions have been used. The reference values a for the *if a then b* rules have been directly integrated to the knowledge base and are interpreted by the workflow-based controller, which can be connected to the data streams of all sensors or feature extraction modules in the system and therefore allows to identify well-defined situations autonomously. In case of more complex situations, external situation detection modules are used, such as the hands-on usage of a robot, where the HMM-based approach of Schreiter [112] is used to perform the situation detection process. Such a detector may be able to detect more than one situation. The output information of every external detector is considered as the interpreted output of a sensor and a reference value a for an *if a then b* rule is modelled in the knowledge base in order to

process the detector output using the workflow-based controller. This allows to compare the detected situation to a characteristic and known output of the situation to be detected.

5.2. Workflow based Controller

As shown in chapter 4 the workflow-based controller is based upon a formal language for the definition of processes, a knowledge base and the controller concept, which is also described in chapter 4. The following chapters will show the atomic concepts used for the workflow management.

5.2.1. Usage of YAWL for this work

The usage of YAWL as shown in chapter 4 is founded in the need for a standardized graphical notation for the definition of surgical workflows for integrated operating theatres. YAWL uses a graphical notation for the definition of processes to allow to share the labour between a process engineer, who is in charge of the process definition and execution and software engineers, who are in charge of the implementation of the actual components that execute an atomic task of a business process. This concept is implemented in YAWL as custom services, which can be registered at the engine. Every atomic task in a YAWL workflow can be connected to a registered custom service to allow for the external execution of an atomic task. In fact, custom services are YAWL's gateways to the actual components that execute the atomic tasks.

In the sense of the shared workload, custom services usually have no knowledge about the state of the process execution, which is entirely processed by the YAWL engine. Custom services are dedicated to a single task, have no knowledge about the overall process and use the variables passed by the engine in order to compute the result of a task execution that is to be passed back to the engine after the atomic task has been finished. The data computed by the custom services, which is fed back to the engine is then interpreted by the engine based on the process' state in order to find the subsequent task(s) of a finished task. YAWL supports the execution of multiple instances of a task as well as the parallel execution of unequal tasks. For the definition of processes YAWL allows to use:

- **Atomic Task:** An atomic task of a process, which is to be executed.
- **Multiple Atomic Tasks:** Simultaneous execution of multiple atomic tasks.

- **Composite Task:** Used to define a sub-process as a sub-net of a YAWL process, whereas a net is YAWL's notation of a process.
- **Multiple Composite Tasks:** Execution of multiple identical sub-processes of a process.
- **Condition:** A condition can be placed between a task and multiple subsequent tasks. The task, which is activated fastest, is executed and all other pending tasks are neglected.
- **Transition:** A transition is used to define the actual control flow in a process. Transitions are directed from a task or a condition to its subsequent elements of the process.

In order to control the flow in the workflow, the process engineer can choose between different split strategies for the activation of multiple tasks that succeed a task. Additionally, YAWL allows to define join strategies, which allow to define when a task with multiple predecessors is activated. If a task has more than one successor, the following split strategies are offered:

- **AND:** Activates all subsequent tasks.
- **OR:** Uses the current state of the process variables to define which subsequent tasks have to be activated.
- **XOR:** Same as OR, but only one task is activated

If a task has more than one predecessor the following join strategies are offered:

- **AND:** All predecessors have to finish before the activation of the task.
- **XOR:** The first predecessor that is finished activates the task.
- **OR:** YAWL evaluates all activated predecessors and activates the task when at least one of the predecessors finished. An OR join is only activated when there are no more active paths that can arrive at the OR join.

For simplification, in this work, Multiple Atomic Tasks, Composite Tasks, Multiple Composite Tasks, Conditions (except input and output condition) and OR splits are not supported by the workflow-based controller and should therefore not be used for the process definition. In contrast to conventional process-based systems, the proposed approach allows to dynamically compose the system that actually executes the tasks of a workflow from knowledge about the situation. As shown in chapter 4, a custom service is used to connect the Master Control to the YAWL engine. As the workflow-based controller is also in charge of selecting components that allow to detect the end of a task the workflow-based controller requires knowledge of the entire process to allow for the selection of these components. The components that can

detect the end of a task and that help the workflow-based controller to find subsequent transitions/tasks are called *transition models* in the following.

The workflow-based controller evaluates the output of each *transition model* and compares it to reference values to compute if a task finished and which task has to be activated next. This information has to be passed back to the YAWL engine. In order to allow for the external processing of subsequent tasks in the workflow-based controller every YAWL process to be used with the workflow-based controller has to follow the design principles described in the following:

- All tasks to be used with the system have to be associated with the custom service that connects to the Master Control.
- Every process includes a net variable (*subsequenttasks*), which is used to keep the a list of subsequent tasks as a string separated by ";".
- The same applies for every task in the process.
- After a task finished, the output mappings of the task copy the payload from the task variable *subsequenttasks* to the net variable *subsequenttasks*.
- For every subsequent task of a task, a predicate is defined which evaluates to true when *subsequenttasks* of the net contains the name of the subsequent task.

Net-variables, task-variables, predicates and output mappings in this scope are built-in mechanisms of the YAWL language. These allow to simplify the implementation of processes for the workflow-based controller. The defined structure allows for the easy generation of processes by a domain expert, meaning that all the constraints made above could possibly be generated automatically by an editor. An editor that supports this structure would therefore have to read all available tasks and transitions from the knowledge base and provide them to the domain expert, in this case the surgical personnel. This allows for the composition of the workflow with little knowledge about the underlying workflow management system.

In order to enable the workflow-based controller to be aware about the entire process, the process is not only loaded to the YAWL engine, but its location in the file system is also stored in the knowledge base. If a workflow is getting started, the location of the workflow is retrieved from the knowledge base, the workflow is loaded and parsed by the workflow based controller as a graph, which is used for the system composition during the execution of the process.

After the workflow has been started by the engine, the engine activates the first task in the workflow. All tasks of a workflow are connected to the custom service implemented in this work. The name of the recently activated task is

getting passed to this custom service, which passes it to the Master Control via ROS. The workflow-based controller then searches for the task in the parsed graph and composes the system dependent on the task's needs and dependent on the transitions leading from the task to all subsequent tasks. This process is depicted in Fig. 5.14

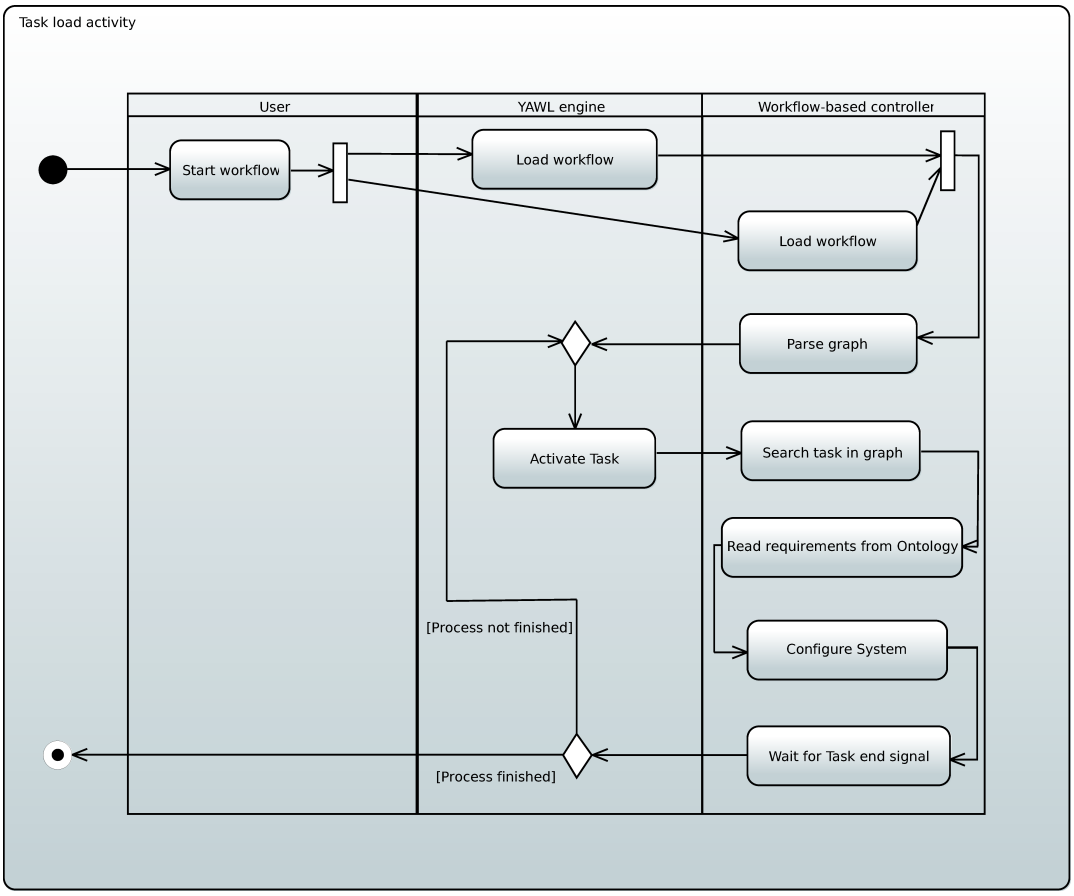


Figure 5.14.: Activity diagram showing the processing of tasks in a workflow.

If a transition and therefore the end of a task is detected by the workflow-based controller, the Master Control signalizes the end of the task to the custom service via ROS and also passes the name of the succeeding task to the custom service, which then passes this information to the YAWL engine. The information about the subsequent tasks to be activated is used to generate the list of subsequent tasks for the task variable *subsequenttasks* of the currently active task that is about to be finished. The YAWL engine then finishes the task and maps the payload of the task's variable *subsequenttasks* to the net variable *subsequenttasks* using the output mappings mechanism of YAWL. Finally, if a task has more than one possible subsequent tasks, the engine evaluates the predicates of the just finished task to select the subsequent task(s) to be activated. The predicates are written as XPath expressions and

for every possible subsequent task, XPath's *"contains"* function is used to evaluate if the list of subsequent tasks in the net variable *subsequenttasks* contains the name of this task. If this is the, case the predicate evaluates to true and depending on the chosen split type of the task, the engine selects the next task(s) to be activated. These steps, which are depicted in Fig. 5.15, are then repeated until the end of the process is reached.

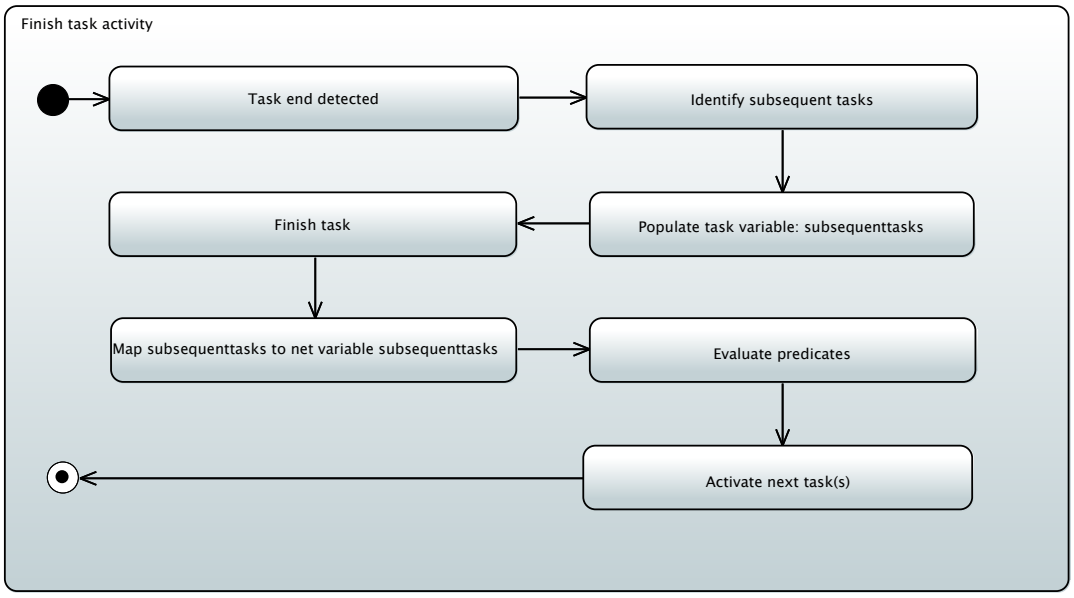


Figure 5.15.: Activity diagram showing the processing of a finished task.

5.2.2. Formalization of surgical workflows in YAWL

Using surgical process modelling techniques and an appropriate amount of observed procedures a representation of the surgical workflow can be created. Such a SPM can be directly represented as an UML activity diagram. The derivation of a YAWL process with the assumptions made in chapter 5.2.1 from a surgical process model can easily be performed. How the elements of an UML activity diagram are translated to the elements of the YAWL language is shown in the following:

- **Initial Node:** The initial node of an UML activity is the analogy to the input condition of a YAWL process.
- **Action:** An action in an activity diagram can be represented using a task.
- **Edge:** An edge is the analogy to a transition in a YAWL process.

- **Decision node:** A decision node can be implemented using a YAWL task with a XOR split, whereas the task is used to interpret the data that is utilized by the decision node for taking the decision about the subsequent task to be activated.
- **Merge node:** A merge node can be implemented using a YAWL task with a XOR join whereas the task does not perform any action and finishes immediately.
- **Fork:** A fork can be implemented by using a YAWL task with an AND split, whereas the task does not perform any action and finishes immediately.
- **Join:** A join can be implemented by using a YAWL task with an AND join, whereas the task does not perform any action and finishes immediately.
- **Activity Final Node:** The activity final node of an UML activity is the analogy to the output condition of a YAWL process.

Using this methodology, one can directly translate UML activity diagrams or SPM to YAWL processes. Such a process however is not yet directly executable by the workflow based controller, as the tasks and transitions are not yet modelled in the knowledge-base of the workflow-based controller.

It is necessary to implement each of the tasks that can occur during a procedure within the workflow-based controller. This may lead to huge tasks that need to be executed and which on their own may have an internal control flow. An example for this is the execution of a laparoscopic procedure, which is desired to be performed using a robotic telemanipulation system nad has also been chosen as relevant application for this work as can be seen in chapter 4.4.

In the laparoscopic telemanipulation scenario, from a surgical point of view the process involves induction of anaesthesia, sterile coverage of the operating field and the robot, sterilisation of the operating field, installation of the robot at the surgical bed, incision of the skin and the abdominal cavity, placement of the trocars, insuflation, the actual treatment performed using bimanual telemanipulation, suturing, and emergence of the anaesthesia.

All of these steps can be described by workflows themself, whereas different granularities can be applied. Also the point-of-view is of great importance for the modelling of the system. From the surgical point of view, the telemanipulation process involves the use of a robotic system to perform the actual steps of the planned procedure, such as the removal of a gallbladder. From a technical point of view, the telemanipulation process involves all steps described in chapter 4.4, such as the calibration process, hands-on approach of the robots or the capturing of the position of the trocars. In order to account

for all technical as well as surgical needs, the final process to be executed by the workflow-based controller is required to include both, technical and medical, aspects.

5.2.3. Modelling System and process knowledge in formal logic

As stated before a consistent knowledge-base is required to allow for the composition of a system configuration from a task of a workflow. This chapter will show the internal structure of the knowledge base which is implemented using OWL-DL [17]. All relevant classes, individuals and properties will be shown. The knowledge in the ontology is divided into process knowledge and system knowledge. As semantic reasoner, to infer logical consequences from the ontology and to connect process and system knowledge as well as to improve the system composition, HermiT [87], which is based on the hypertableau algorithm is used.

5.2.3.1. Process knowledge

The base class for all process-related knowledge is the *WorkflowElement*. Every Workflow, its tasks and its transitions are derived from *WorkflowElement*. The classes *Task* and *Transition* are the atomic elements of a workflow and are therefor derived from *AtomicWorkflowElement*, which is derived from *WorkflowElement*. The structure of the process-related knowledge in the ontology can be seen in Fig. 5.16. It is not explicitly modelled, which tasks and

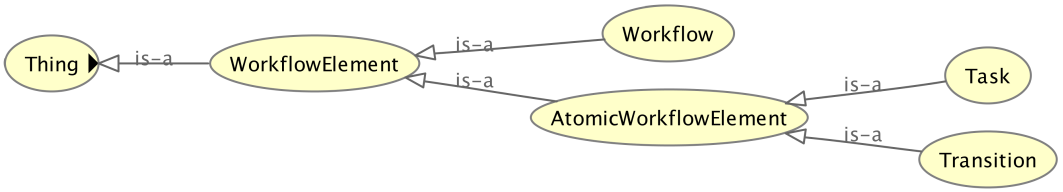


Figure 5.16.: The classes representing the workflow related knowledge as modelled in the ontology

transitions are part of a workflow. This means that tasks and transitions are assigned to a workflow during the modelling process and can be reused in other workflows.

When a user requests to execute a workflow, the workflow-based controller reads the associated individual of type *Workflow* from the knowledge base. Each individual of type *Workflow* is assigned to a location in the file system

where the described workflow is stored. As shown in chapter 5.2.1, the workflow-based controller parses the YAWL file. The graph from the YAWL file is then used to assign the tasks included in the YAWL file to the tasks modelled in the ontology. However, this means that for every task that may occur in a workflow an associated individual of the type *Task* has to be modelled in the ontology. In case a workflow is executed, which includes tasks that are not modelled, the execution of the workflow will fail as soon as this task is reached. Ontologies that do not include all tasks of a workflow that is getting executed are considered to be inconsistent. Additionally, for every task modelled in the ontology, the ontology contains at least one individual of type *Transition* that is connected to this task. Using this/these Transition(s) the workflow-based controller is able to decide which *transition models* are suitable to detect the end of a task and to detect which transition has to be activated.

Using this methodology, a task and the associated transitions are not defined as parts of the workflow to be executed but as atomic workflow elements that can be utilized by any workflow to be executed by the workflow-based controller. Moreover, this means that all tasks and transitions modelled in the ontology can be used for the composition of a workflow and therefor these atomic workflow elements can easily be reused in case the process changes or in case an entirely different workflow is modelled.

The Master Control in this scope will wait until the workflow engine activates a task and passes this information to the Master Control via the custom service. The Master Control will then search the task in the graph that has been parsed using the workflow file. Afterwards, the Master Control queries the ontology for this task and if the query is successful, the ontology is queried again for all transitions leading from the task to its subsequent task. In case the task and all of its transitions are found, the Master Control will try to compose the system, which is performed using requirements of tasks as well as events that are significant for the activation of transitions and therefore for the end of the task. These requirements and events are modelled in the ontology for every task and every transition.

5.2.3.2. System knowledge

The system knowledge is comprehensive compared to the process knowledge. System components are modelled in the ontology using the base class *Component*. From this class the classes *RosComponent*, *Device* and *Driver* are derived, whereupon all components that are implemented as ROS components of the system are modelled as individuals of the subclasses of *RosComponent*. All physical devices such as robots, optical tracking systems, or cameras are

modelled as individuals of class *Device*. The structure of component-related knowledge as modelled in the ontology can be seen in Fig. 5.17

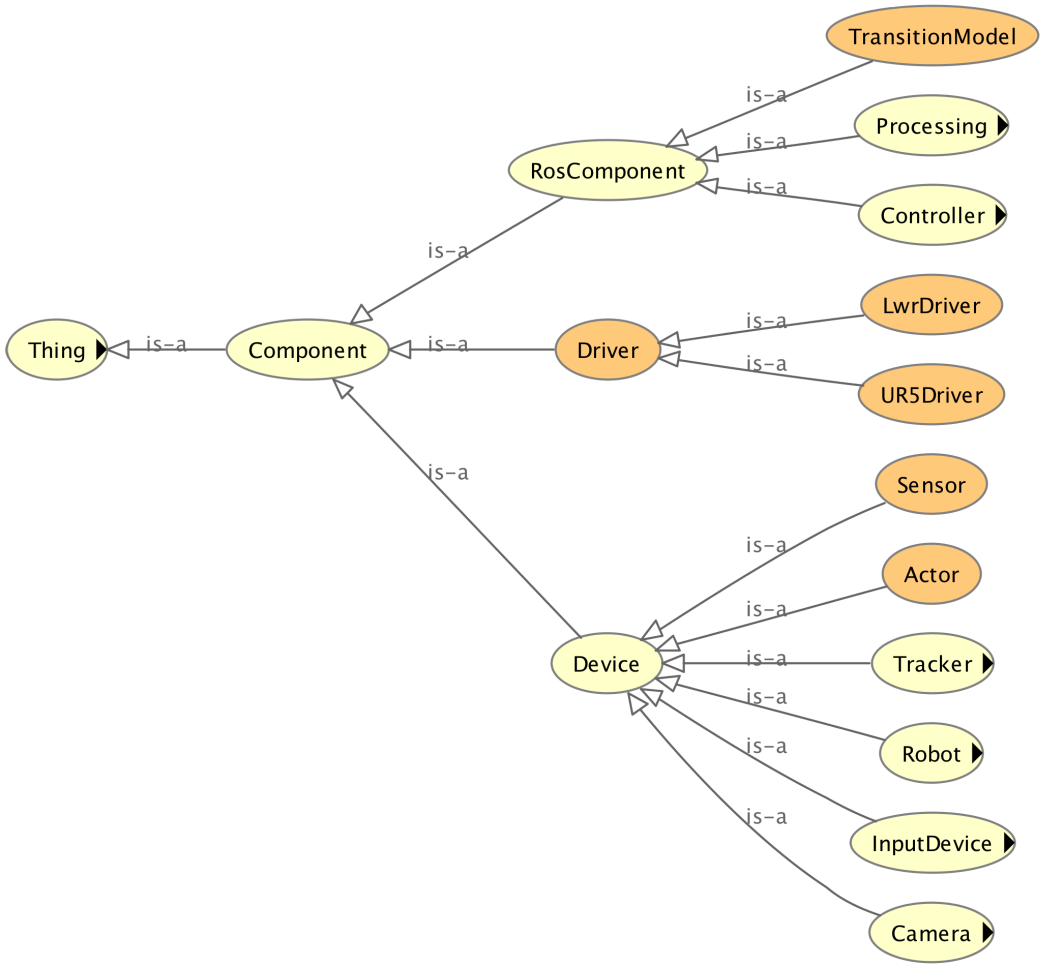


Figure 5.17.: The classes representing the component related knowledge as modelled in the ontology. Yellow: Primitive classes, Orange: Defined classes

As can be seen, a *RosComponent* can be of type *Controller*, *Processing* or *TransitionModel*. *TransitionModel* in this scope is a defined class. This means that instead of explicitly modelling a *RosComponent* to be a *Transition Model*, the reasoner is used to classify any *RosComponent*, which is able to detect a transition as a *TransitionModel*. Every component that is classified as a *TransitionModel* can be chosen by the workflow-based controller in order to detect a transition and therefor to detect the end of a task. The actual selection process for suitable transition models from individuals of type *TransitionModel* is shown in chapter 5.2.4. All individuals representing a physical component of the system, which are connected to the system through an additional *RosComponent* are modelled in subclasses of *Component*.

The category of *RosComponent* is further divided in components of class *Controller* and components of class *Processing*. An individual of type *Processing* usually is not directly connected to a device. Examples for individuals of type *Processing* are the components that perform the fusion of the point clouds of the camera system. In contrast to this, individuals of type *Controller* are required by one or more individuals of type *Device*, to make them available via ROS. In this scope, the defined class *Driver* is used to classify all controllers for individuals of class *Device*. Therefor the class *Driver* is further divided into *LwrDriver*, which contains all individuals of type *Controller* that can serve as *Driver* for the LWR4 (KUKA Roboter GmbH, Augsburg, Germany), and *UR5Driver*, which contains all individuals of type *Controller* that can serve as *Driver* for the Universal Robots UR5 (Universal Robots, Odense, Denmark).

In order to allow for the interpretation of data, which flows from the components to the workflow-based controller and to allow for the configuration of system components, the ontology also has to contain knowledge about the transmitted data. This knowledge is collected using the class *ControlStructure*. As can be seen in Fig. 5.18. *ControlStructure* is further divided into the classes *ComponentOutput* and *RosControlStructure*. Individuals of type *ControlStructure* are representations of the mechanisms used to transmit data between ROS nodes (modelled as individuals of type *RosComponent*) using ROS services, ROS topics and *dynamic_reconfigure*. In this scope, it is further differentiated between *ReconfigurableParameter* and *RosDataFlow*, where reconfigurable parameters are used to change the behavior of components and Topics as well as Services, collected under *RosDataFlow*, are used to transmit data between components. In order to allow for the utilization of individuals of type *RosControlStructure*, each of them is described by a name (address in the system) and a data type, which allow to access and to provide information using the *ControlStructure*.

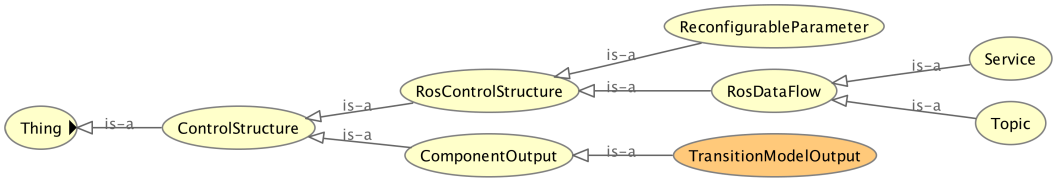


Figure 5.18.: The classes representing the knowledge about transfered data in between system components. Yellow: Primitive classes, Orange: Defined classes

Individuals of type *ComponentOutput* can be seen as a representation of data as it flows in the control system. It represents a set of data as it can be generated by one of the individuals of type *RosComponent*. Each individual of type *ComponentOutput* is related to a *RosDataFlow* on which data is expected

to arrive and to a value for the data that is expected to be generated by the associated *RosComponent*. An individual can therefor be used to compare the expected value from the ontology with actual values generated by the system. The types of data are modelled as they are used in the actual execution system. For ROS services, all types of data as used in ROS *std_srvs* are allowed. For ROS topics all types of data as used in ROS *std_msgs* are allowed. The representation of the available types of data is depicted in Fig. 5.19

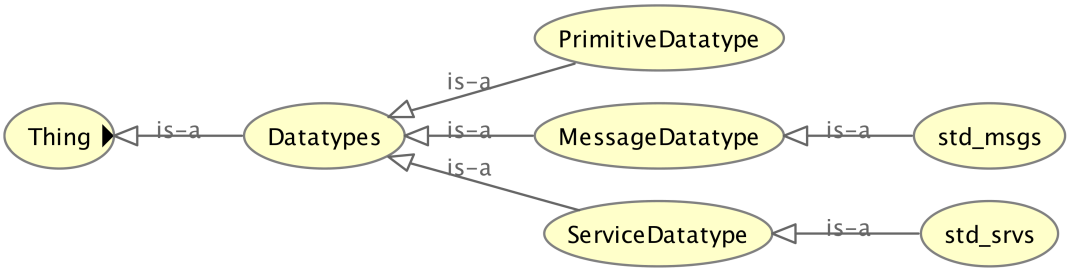


Figure 5.19.: The classes representing the knowledge about the types of data used.

This enables the workflow-based controller to interpret data, which is distributed between system components. In order to allow for the configuration of components, the *dynamic_reconfigure* mechanism of ROS is used. For this purpose, individuals of type *ReconfigurableParameter* are used. In the ontology it is defined for which component a *ReconfigurableParameter* can be set. An actual configuration for a *RosComponent* is represented by an individual of type *ComponentConfiguration*. Such a *ComponentConfiguration* includes a set (at least one) of individuals of type *ParameterConfiguration*. A *ComponentConfiguration* is related to one ore more individuals of type *RosComponent*, which can be configured using the configuration. Each *ComponentConfiguration* is defined to be suitable for a task that is modelled in the ontology. A *ParameterConfiguration* expresses an actual value for a *ReconfigurableParameter*. The collectivity of *ComponentConfiguration*, *ParameterConfiguration* and *ReconfigurableParameter* therefore allows to actually configure one or more individuals of type *RosComponent* for the task that is pending to be executed. *ParameterConfiguration* and *ComponentConfiguration* are specializations of *Configuration* as can be seen in 5.20

Using this knowledge, it is still not possible to build a system composition/-configuration from the process knowledge. Additionally, individuals of class *Event* are required. An event in this scope is defined as a trigger, which signalizes that a transition has been activated and therefore that the task that has been executed prior to this transition has ended. In the ontology, individuals of type *ComponentOutput* can be defined to signalize the presence of an event to which the *ComponentOutput* is associated. If data, which matches the reference value defined for the *ComponentOutput* occurs on the *RosDataFlow*,

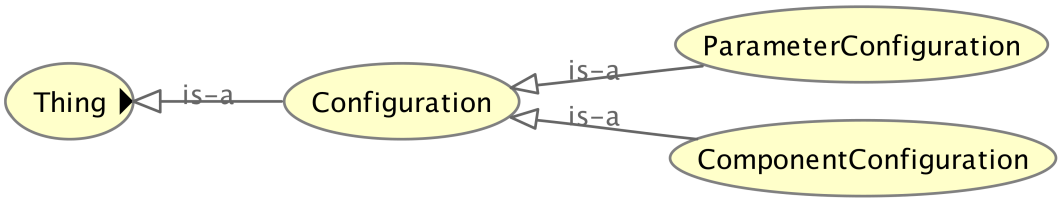


Figure 5.20.: The classes representing the knowledge about configurations for components

connected to the *ComponentOutput*, the workflow-based controller assumes that the *Event* connected to the *ComponentOutput* has been activated. Using the defined class *TransitionModelOutput*, the reasoner will classify individuals of type *ComponentOutput* that are connected to an *Event* as individual of type *TransitionModel*. Using the information that a *ComponentOutput* is a *TransitionModelOutput* and that a *ComponentOutput* is connected to a *RosComponent*, the reasoner automatically classifies all individuals of type *RosComponent* that can detect a *Transition* as individual of the defined class *TransitionModel*. This completes the knowledge needed to select a suitable *RosComponent* for the detection of a transition in the workflow, as for every transition the associated events are known. In terms of reusability an event can then be used to signalize more than one transition and a *RosComponent* can detect more than one event. The definition of a new event therefore only requires to define a *ComponentOutput* that includes a reference value on a *RosDataFlow* that is characteristic for the output of a *RosComponent* that is generated when it detects the desired event (i.e., the dead man switch for a telemanipulation system sends *true*, which equals the reference value for the event *telemanipulation_activated_event* that initiates the transition to *telemanipulation_bimanual*).

In order to connect a task to an execution component, individuals of type *Task* can be defined to require one or more individuals of type *Component*. This relation is transitive, which means that a required component may also require other components, which then can be found by the reasoner recursively. This defines a set of individuals of type *Component* that are required to execute a task (i.e. a task requires a LWR4, which requires a LWR4 controller). In case a component can be replaced by another component, the replacing component can be defined as an alternative to the component that is to be replaced. This relation is also transitive, which allows to create alternative system configurations for a task in case a required component is dysfunctional or missing.

All relevant classes of the ontology including the *Event* class are shown in Fig. 5.21

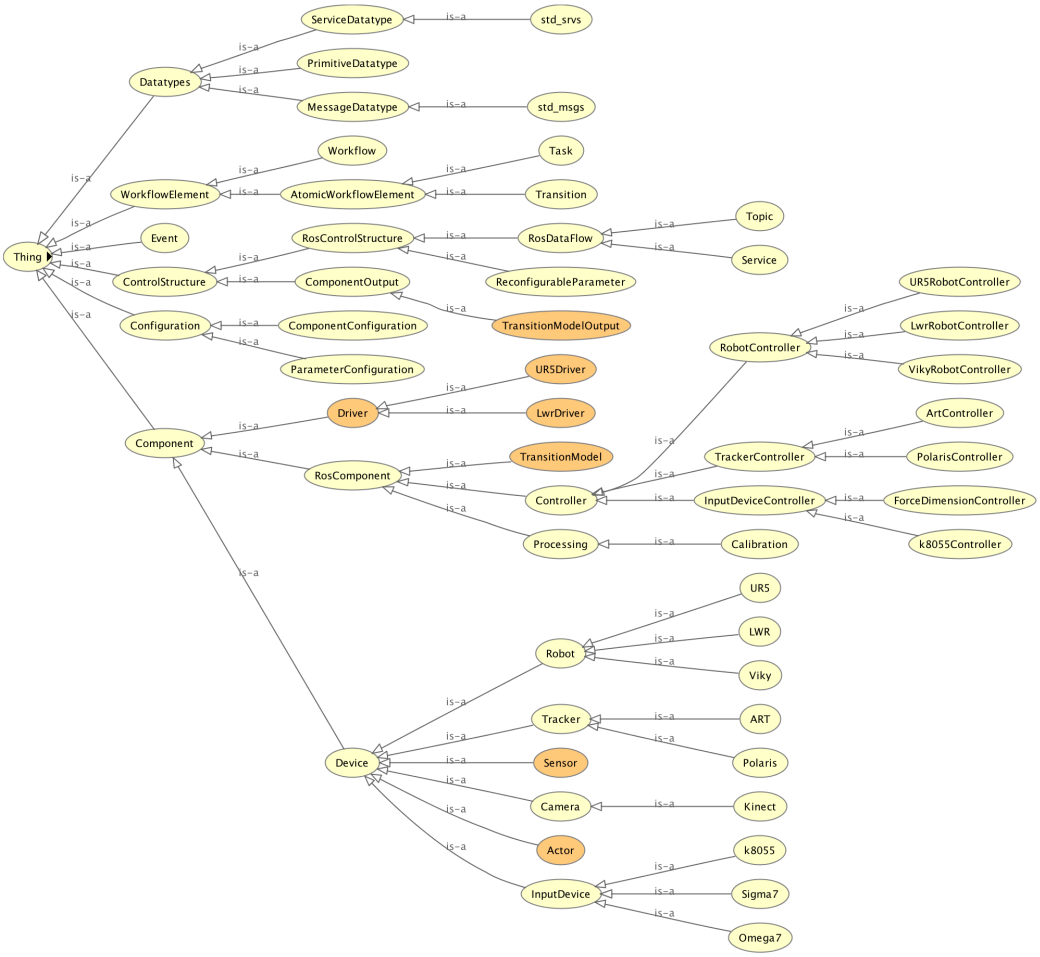


Figure 5.21.: The relevant classes of the ontology representing both, system and process knowledge. Yellow: Primitive classes, Orange: Defined classes

5.2.4. Composition and Configuration of the execution system

As stated before, the workflow-based controller consists of three components. These are the YAWL engine, the Custom Service and the Master Control. The custom service is implemented as a servlet and deployed to the same Apache Tomcat Container in which the YAWL engine runs in. In order to connect to ROS, the Custom Service utilizes the ROSJAVA libraries. All information that has to be passed from the YAWL engine to the Master Control is transmitted as ROS service calls via the Custom Service, which is also used to receive ROS service calls from the master control to be passed to the YAWL engine.

The Master Control is implemented as a regular ROS node and is written in Python in order to be able to utilize services, topics, dynamic reconfigura-

tion and the ROS Master API. However, the capabilities of python and the capabilities of available external libraries are limited in terms of knowledge processing. In order to allow for easy access to the data of the ontology, reasoner support and support for SPARQL-DL [117], the knowledge processing is performed in JAVA for which comprehensive knowledge processing tools exist. The python-based parts of the master control are in charge of querying the execution components, passing data to the execution components through services, topics and *dynamic_reconfigure*, as well as for the sniffing of data from the system in order to allow for the interpretation of this data to detect transitions.

The JAVA-based parts of the Master Control are in charge of the knowledge processing and for the control of the python-based parts of the Master Control. For this purpose, the JAVA-based parts are compiled as a library. The library is then used by the python-based parts via JPype, where the python code accepts requests from the engine and passes these requests to the JAVA-based parts, where they are processed using the ontology. The result is passed back to the python-based parts, where the topic subscribers and service servers are configured, the required nodes are queried using the ROS master API and the data from the execution components is compared to the reference values retrieved from the ontology. Before any knowledge from the ontology is processed, the HermiT reasoner is used to precompute inferences in the ontology. The resulting inferred model is then queried through SPARQL-DL, which is a query language for OWL-DL ontologies. The SPARQL-DL requests are processed by HermiT. The results from the query are therefor derived from the inferred model and can contain automatically classified individuals and classes of the ontology.

Using this architecture, a workflow is executed as follows:

1. The workflow-based controller is started with reference to the desired workflow.
2. The workflow-based controller classifies the ontology using the HermiT reasoner and retrieves the location of the workflow in the filesystem.
3. The workflow is loaded from the filesystem and parsed as a directed graph.
4. The workflow-based controller triggers the YAWL engine to execute the workflow through the custom service.
5. The YAWL engine executes the workflow and activates the first task(s) of the workflow.
6. The information about the task is passed to the workflow-based controller through the custom service.

7. The workflow-based controller retrieves information about the transition models and the required/alternative components.
8. Using this information, a suitable system configuration is set up.
9. The workflow-based controller configures all the components of the system configuration to fit the needs of the current task.
10. The workflow-based controller sniffs the data, transmitted by transition models, and compares the data to the reference values as retrieved from the knowledge base.
11. If a match of the sniffed data and the reference value for a transition occurs, the workflow-based controller communicates the task's end and the next task to be activated to the YAWL engine via the custom service.
12. The YAWL engine activates the next task and the control flow continues at step 6. This is repeated until the end of the process.

For the system composition, the workflow-based controller queries the ontology for the task itself. If it is contained in the ontology, the ontology is queried for all required components of this task. Due to the use of a reasoner and the transitivity of the *requiresComponent* property of a task or a component, the workflow-based controller can guarantee that all of the requirements for every component and task that are used, are identified. After the identification of each required component the ontology is queried for drivers that may be required by a component but which are not defined as necessary components. This approach allows to connect a component directly to its driver in contrast to the loose coupling via the *requiresComponent* property. After this step, the set of components to execute the task is known. It is not yet known, whether one of the components is unavailable and how the components have to be configured for a suitable system configuration. For this purpose, the ROS Master API is used to query each of the component for availability. This is performed using the ROS master, which, as a central component, is aware about the address of each running component. If a component does not respond, it is considered to be *unavailable*, which means that it is required to be replaced by an alternative component. Therefore for each component that is considered to be *unavailable*, the ontology is queried for an alternative. Alternatives can be defined for each component in the ontology individually. If such an alternative exists, it is retrieved from the ontology and is queried for availability. If this component is also not available, other alternatives for a node can be selected and queried. Also alternatives of alternatives can be searched recursively until a suitable component is found. If no alternative can be found or none of the alternative components is available, the task cannot be executed. This methodology assures that all components required for task execution are in place and running.

In order to allow for the detection of the task' transitions and therefore for the end of a task, the ontology is queried for all transitions that are subsequent to a task. A task is not modelled as part of a specific workflow but as a generic component that can be used in any workflow. To allow for the identification of task transitions present in the executed workflow, the graph computed from the workflow to be executed by the engine is searched for the transitions of a task as found in the ontology. If the transition is found in the graph and in the ontology, it is considered to be executable. This underlines the importance of a consistent ontology as for every transition in a process a transition and a suitable behavior have to be modelled in the ontology in order to correctly execute a process. Using the knowledge representation as shown in chapter 5.2.3 it is directly possible to query the ontology for transition models that are suitable to detect a given transition. For every transition of the task to be executed, the ontology is therefore queried for suitable transition models. The workflow-based controller then queries the transition models for availability and responds with an error, if a model to detect a transition is not available and no alternative can be found. For every transition model the ontology is then queried for the events that can be detected by the transition model and that are characteristic for the transition to be detected by the transition model.

As can be seen in chapter 5.2.3, the workflow-based controller expects a characteristic output of a transition model for every event that can be detected by the transition model. This transition model output is retrieved from the ontology for every event relevant for the detection of the transition. For this transition model output, the ontology is queried for the address (e.g. topic, service), where the output data of the transition model is expected on, as well as for the type of data, which is expected on this address. Finally, the reference value to which the data on the address is compared to is queried from the ontology. This information is then used to subscribe to the ROS topic or to advertise a ROS service server associated to the retrieved address. After this step, the workflow-based controller receives all data advertised on this topic or transmitted by a service client and compares the retrieved data with the reference value from the ontology. If a match is found, the associated transition is activated, the associated task is finished and the next task to be activated, which is known from the process file, is transmitted to the YAWL engine.

The complete list of components to be activated for a task includes all required components for the execution of the task and the components that are transition models for the transition detection/task ending detection. If a suitable set of components could be successfully created the system composition is finished. At this point, all execution components, necessary for the execution of the activated task, are running and the workflow-based

controller is configured to check the output data of the transition models for a characteristic value/event that signalizes the end of a task.

In order to allow for the execution of the task an additional step, the system configuration, is needed in order to correctly configure the components for the task to be executed. During the system configuration, components are configured to build pipelines between each other. Such a pipeline can e.g. be an imaging pipeline or a pipeline for the transmission of positional data from an input device to the robot. For this purpose, the components are configured to subscribe to topics or to publish on topics known from the ontology, which allows to dynamically link the components. The configuration process also includes the configuration of parameters of the components to optimize components for the task and to enable or disable components. This e.g. includes the configuration of thresholds or algorithm parameters.

For configuration purposes, the ontology's *ComponentConfiguration* class is utilized. For each component to be used, the ontology is queried for a *ComponentConfiguration* that is suitable for the component and the task. If such a configuration is found in the ontology, the elements of this configuration (*ParameterConfiguration*) are retrieved from the ontology, which contain values for reconfigurable parameters of the associated component. For every *ParameterConfiguration*, the associated reconfigurable parameter is then retrieved from the ontology. This enables the workflow-based controller to dynamically reconfigure a ROS node during run-time using the ROS *dynamic_reconfigure* mechanism. In this scope the workflow-based controller is not aware about the implementation of the reconfigurable parameters on the component side. The responsibility of the processing of the requests to change a parameter by the workflow-based controller is entirely given to the component that is to be configured. This allows to generically implement the knowledge about the reconfigurable parameters of a component in the ontology without knowledge about the actual implementation on the component side.

A reconfiguration of a parameter can invoke the following actions on the component side

1. activate or deactivate a function of a component or the entire component
2. change the parameters of an algorithm used within a component
3. change of a data flow/pipeline within the system

In this scope, action 3 allows for a dynamic data flow composition. This allows to exchange components of processing pipelines during run time in order to adapt the pipeline to the task or to replace components that failed for any reason, during or before the execution of a workflow.

A data flow composition can either be static, where the components are aware about data sources and target before run time, or dynamic, where the sources

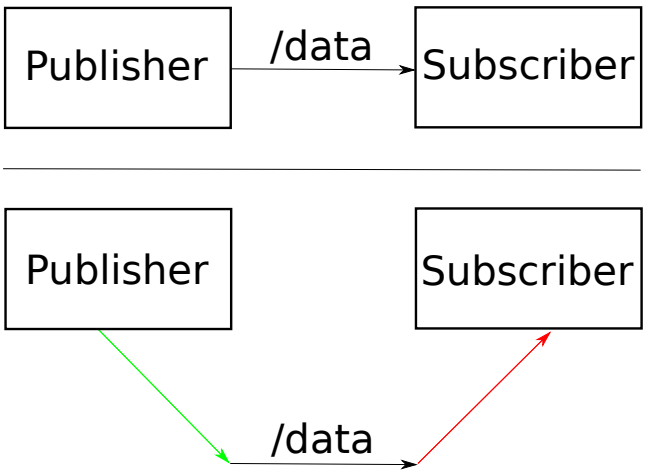


Figure 5.22.: Top: Static system composition, where the subscribing and the publishing node require the address of the topic/data before run time. Bottom: Dynamic system composition. The address of the topic data is provided to the publishing and subscribing node during run time. The publisher publishes to the topic data after the address has been provided to the node (green arrow) and the subscriber subscribes to the topic after the address has been provided to the node (red arrow).

and destinations can be changed during run time via the reconfiguration of a reconfigurable parameter. Also a combination of both is possible. A dynamic data flow composition therefore allows to change links between components based on the needs of a task. The difference between a static and a dynamic data flow composition is shown in Fig. 5.22.

If dynamic composition is chosen for a link between two components, it has to be taken care that the knowledge in the ontology is modelled properly as in case of inconsistencies, data may be unvailable to components or may be provided on wrong addresses (e.g. topics, service), which may lead to unexpected behavior or a failure of the entire system. In order to assist this process, for every data available in the system (such as number of people detected, pedals pressed, robot positions) a dedicated unique address is defined and each component that provides or retrieves data from/to this address is configured to subscribe to, or to publish on this address.

The system composition and configuration is shown as pseudo code in algorithm 1.

Algorithm 1: System composition and configuration by the workflow-based controller

```
Data: task
activateTask;
componentlist=read components for task from ontology;
get all transitions of task from task graph;
get finish events for transitions from ontology;
get TransitionModels for each event from ontology;
add TransitionModels to componentlist;
foreach component ∈ componentlist do
    check Availability of Component;
    if component unavailable then
        get alternative components to component;
        check alternatives for availability;
        if no alternative found then
            fail;
        end
    end
    retrieve suitable ComponentConfiguration for component and task
    from ontology;
    configure component with all ParameterConfigurations from
    ComponentConfiguration;
    if component ∈ TransitionModel then
        start listening for the output of the TransitionModel;
    end
end
while task running do
    foreach activated TransitionModelOutput listener do
        compare TransitionModelOutput with associated reference
        value from ontology;
        if reference value matches TransitionModelOutput then
            finish task;
            send target task for fired event to YAWL engine;
            stop listening for other outputs of TransitionModels;
        end
    end
end
```

5.3. Execution of a laparoscopic test case

The methods show in the previous chapters have been applied to the laparoscopic scenario using the workflow and the tasks described chapter 4.4. The application scenario has been analyzed and a suitable workflow that can be executed using the workflow-based controller has been designed.

The assumption has been made that the workflow starts with an uncalibrated system, which is brought to the operating room after or before the patient is placed on the operating bed. The system therefore has to be calibrated in the first steps, then the system has to be equipped with surgical instruments, which may be exchanged during the operation. After that, the laparoscopic tools as well as the endoscopic camera have to be introduced via trocars for which the hands-on mode of the surgical robots is used. Then the actual telemanipulation can take place, whose implementation is based on a master-slave robot system, where the surgeon performs the operation using haptic input devices and where the robots perform the actions on the patient side. After the operation, the personnel extracts the instruments that are still attached to the robot from the patient using the hands-on mode.

From this workflow, several medical and technical tasks have been extracted, which are shown in the following.

- **calibrate_robot_1:** A purely technical task, which localizes robot number 1 (left hand) in the scene using the optical tracking system and the calibration algorithm shown in chapter 5.1.1.4. The task is finished as soon as the robot has performed the calibration and moved to a preplanned target pose.
- **calibrate_robot_2:** The calibration of robot number 2 (right hand) is performed using the procedure from the previous step.
- **capture_trocar_1:** A combined technical and medical task in which the location of the trocar point for the robot that is equipped with the tool for the left hand of the surgeon is selected by the surgical personnel. After the selection and placement, the position of the trocar for the surgical instrument attached to robot 1 is captured using the NDI pointer and the optical tracking system. The robotic system uses this trocar as remote center of motion during the operation. The task is finished as soon as the operator acknowledges the capturing of the trocar using a button on a graphical user interface.
- **capture_trocar_2:** The position of the trocar for the robot equipped with the tool for the right hand of the surgeon is selected using the procedure from the previous step.

- **robot_start_pose_1:** A convenience task that moves the robot with the tool for the left hand of the surgeon to a predefined starting pose. This starting pose is ergonomically designed to allow to attach a surgical instrument to the robot. The task is finished as soon as the robot reaches its start position.

- **robot_start_pose_2:** Moves the robot with the tool for the right hand of the surgeon to a predefined starting pose. The task is finished as soon as the robot reaches its start position.

- **move_camera_start_pose:** The personnel moves the camera carrier to the desired starting pose. The camera carrier used is the Viky camera steering robot (EndoControl, Grenoble, France), which can be controlled by voice or a footpedal and which can be initially moved by a human operator in order to attach it to a trocar. The personnel also mounts the surgical instruments to the flanges of the robots. The task is finished as soon as the operator acknowledges the end of the installation process using a button on a graphical user interface.

- **camera_in_start_pose:** The camera carrier is at the starting position and the system is waiting for human interaction. The task is finished as soon as the system detects the desire of a human operator to move one of the robots in hands-on mode.

- **hands_on_mode_robot_1:** The left hand robot is used in hands-on mode. The task finishes as soon as a human operator stops touching the robot's end effector.

- **hands_on_mode_robot_2:** The procedure of the prior step is performed for the right hand robot.

- **fixed_mode_robot_1:** The left hand robot is fixed and will not move. The task finishes as soon as the operator starts the telemanipulation process by using the dead man pedal, or when the operator's desire to move one of the robots in hands-on mode is detected.

- **fixed_mode_robot_2:** The procedure of the prior step is performed for the right hand robot.

- **telemanipulation_bimanual:** A bimanual telemanipulation task with two robots, an endoscopic camera and two input devices is performed. In this task, the actual surgical procedure is performed by the surgeon using the master-slave telemanipulation system. The task ends as soon as the desire of the operator to use one of the robot in hands-on mode is detected.

- **trocar_inside_outside_detection_1:** A purely technical task in which the system detects if the surgical tool attached to the left hand robot is still inside the patient's body or not. The task ends if the operator continues to telemanipulate, if the system detects the operators desire to use the robot in hands-on mode, or if the tool attached to the robot is outside of the patient's body.

- **trocar_inside_outside_detection_2:** The procedure of the prior step is applied to the right hand robot.

- **robot_finish_pose_1:** The left hand robot moves to the preplanned finish pose in which the personnel can demount the robotic system from the surgical bed. The task finishes when the robot reached its finish pose.

- **robot_finish_pose_2:** The procedure of the prior step is repeated for the right hand robot.

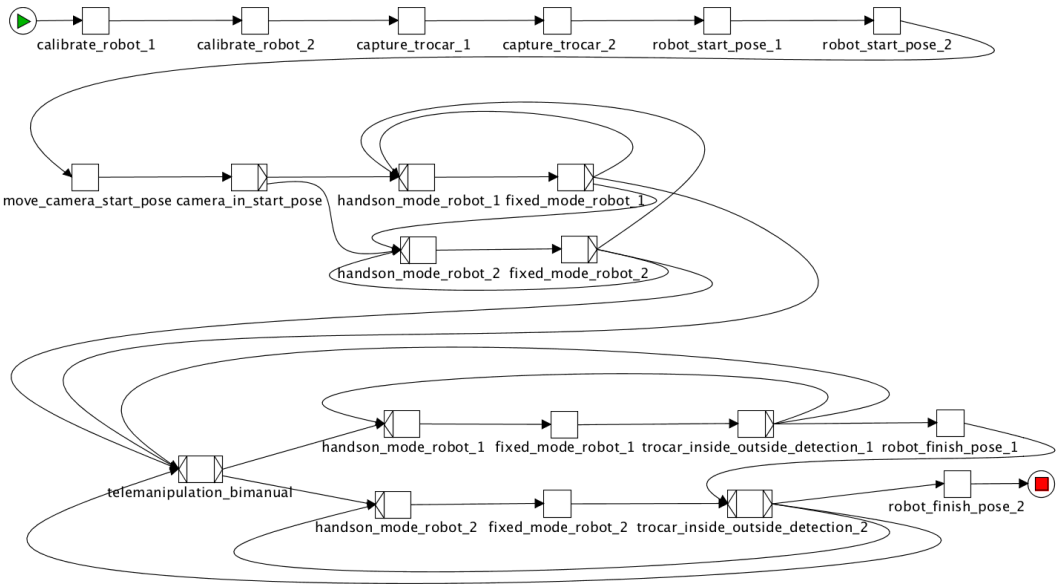


Figure 5.23.: The experimental workflow used in this work, composed of autonomous, hands-on and telemanipulation tasks as well as calibration tasks

Tool changes and recalibration tasks were not foreseen in the test workflow. This means that the personnel can relocate the surgical tool during the operation using the hands-on mode but as soon as the personnel extracts the tool from the patient, the associated robot will drive to its finish pose. For a surgical procedure, where tool changes are necessary the workflow can be easily modified to detect a hands-on mode in the finish pose, which equals the start pose and which allows to exchange the instruments. This modification

would allow to reintroduce the tool attached to the robot through the trocar and to continue the operation using a different surgical tool.

The transitions of the task are modelled in a way that the system has to be calibrated first. Afterwards the surgical personnel at anytime can relocate each of the robot in hands-on mode to adapt the position of the surgical tools to the needs of the procedure or the operating surgeon. This can also be done during telemanipulation.

The entire workflow with all transitions can be seen in Fig. 5.23. During the execution, the personnel is provided with a visualization of the workflow, which is generated from the parsed workflow file by the Master Control. This visualization is displayed on a 40 inch television screen during the procedure and helps to navigate during the execution of the workflow. The visualization can be seen in Fig. 5.24

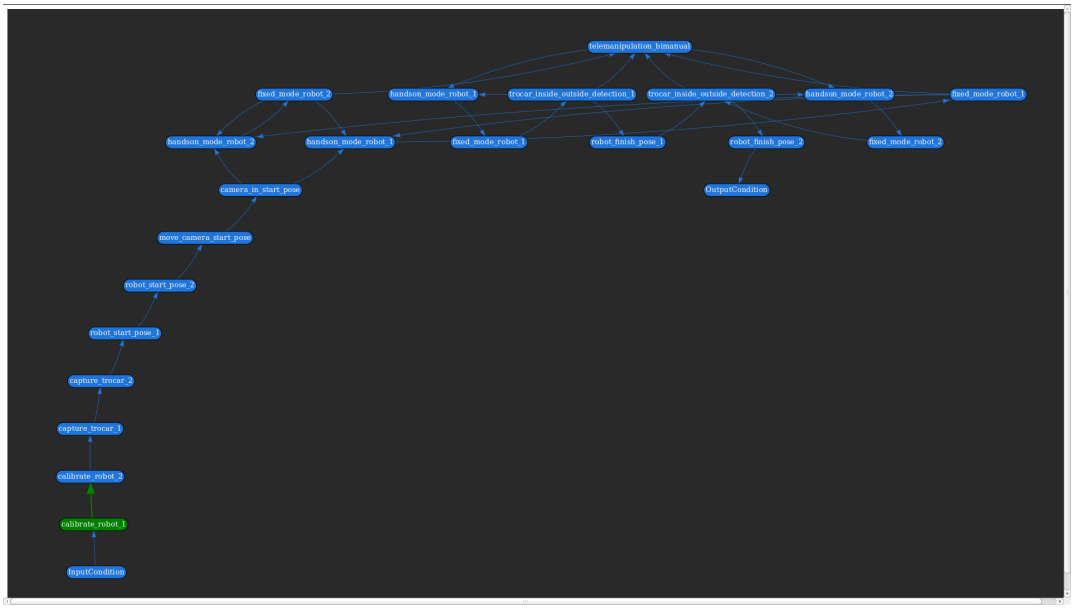


Figure 5.24.: The intraoperative visualization of the workflow. Active tasks and transitions that can be followed after the execution of the active task are marked in green

It can be seen that in the workflow different granularities are used. The focus of this workflow was to ease the usage of a telemanipulation system in the operating room. If it is the desire of the surgeon to be assisted during the execution of the laparoscopic task using the telemanipulator an analysis of the the tasks that happen during the laparoscopic telemanipulation would be required. Examples for such tasks may be cauterization, cutting or suturing. A suitable element to switch between different granularity levels are YAWL's Composite tasks, where tasks modelled as Composite tasks remain

present as a tasks in the workflow but include a subprocess, where the actions during a task can be modelled as a workflow as well. In the test scenario it is assumed that the telemanipulation system is used by the surgeon as is without the introduction of knowledge about the subtasks performed during telemanipulation.

The following paragraphs shows the execution of the workflow. This includes the description of the actions performed by the system, the structure of the components used and the knowledge used, as modelled in the ontology. The transitions of every task are described in the paragraphs and can also be seen in Fig. 5.23. The entire workflow and the actions to be taken are purely modelled by the workflow and the knowledge as modelled in the ontology. None of the connections between the workflow and the system are hard-coded.

5.3.1. Robot calibration

The tasks *calibrate_robot_1* and *calibrate_robot_2* are intended to find the robot base position in the coordinate frame of the optical tracking system, which is used as global reference for the telemanipulation system. The task *calibrate_robot_1* is the first task, which is started during the execution of the workflow using the workflow-based controller. Therefore the engine will be commanded to execute the workflow after the Master Control parsed the workflow as a directed graph. When the task *calibrate_robot_1* is getting started the Master Control searches the graph for the task and will find it as it is the only direct child of the InputCondition of the workflow.

From the graph, the Master Control can also find the transitions leading to successors of *calibrate_robot_1*, which in the workflow is only task *calibrate_robot_2*. The Master Control now queries the ontology if there is a transition modelled that leads from *calibrate_robot_1* to *calibrate_robot_2*. If this is the case the ontology is consistent for the task *calibrate_robot_1*.

Afterwards the Master Control queries the ontology for components that are required for the execution of the task. For task *calibrate_robot_1*the required components are the components *art_1*, *lwr_1* and *robot_pose_estimator_1*. *lwr_1* is a KUKA LWR4, which is identified by number 1 and is used as the left hand robot and *art_1* is the optical tracking system. Both of them are of type *Component* and are defined to require a driver of type *RosComponent* to facilitate the usage of the components via ROS. The remaining component *robot_pose_estimator_1* is a ROS component, which performs the actual calibration process by moving the robot in the field of view of the tracking system and by calculating the transformation between the optical tracking system frame and the robot's base frame. Then for each component it is

checked, using the reasoner, if further components are required. In this case this applies for all three components as the LWR4 and the ARTTrack require a driver and the robot pose estimator is defined to require the ARTTrack and the LWR4 with identifier 1. This adds some redundancy as the *lwr_1* and *art_1* have been explicitly modelled to be required by the task and are automatically selected to be required by *robot_pose_estimator_1*. The selection process ends with a list of individuals of type *RosComponent* that were selected for the execution of a task. In this case the list contains the controller/driver for the ARTTrack system, the controller/driver for the LWR4 robots and the robot pose estimator. For all of these components no alternative has been defined, which means that if one of the components fails, the task can not be executed.

In the next step the workflow-based controller retrieves the unique identifier of the components, with which they register at the roscore and queries each of the components by pinging them. If the components answer, they are considered to be *available*. If one of the components fails in this step the task can not be executed as no alternatives have been defined for the components used in this task.

When a set of components that can execute a workflow is found, the Master Control queries the ontology for all individuals of type *TransitionModel* that can detect the end of a task. For each of the events (in this case one) that are defined to signalize the activation of a *Transition* of a *Task*, a suitable *TransitionModel* has to be found. The Master Control finds a suitable *TransitionModel* for a *Transition* by searching a *TransitionModelOutput* that signalizes the *Event* for a *Transition*. This is possible as a *ComponentOutput* that detects a *Transition* is automatically classified as *TransitionModelOutput* and as a *ComponentOutput* is associated to a *RosComponent*. This means that a *RosComponent* that can output a *ComponentOutput* of type *TransitionModelOutput* is classified as a *TransitionModel* and can therefore be found for the specific *Event* and *Transition* using the reasoner. For the task *calibrate_robot_1* the component *robot_pose_estimator_1* is found as a *TransitionModel* as it can signalize that the calibration is finished. The Master Control then retrieves the topic/service as well as the data type and the value (which is the reference value for comparison) for the associated *ComponentOutput*/*TransitionModelOutput* and builds up the "control loop" as shown in chapter 4.2. From that point on the Master Control listens for the data on the topic/service and compares it to the reference value. If it matches the event occurred, the task is finished and the Master Control can signalize, which transition has been activated and can pass this information to the YAWL Engine. In the case of *calibrate_robot_1*, the Master Control will compare the output data of the component *robot_pose_estimator_1*, which is transmitted on a topic with the reference value *true*. As soon as the reference value occurred on that topic the task is finished and the workflow proceeds with *calibrate_robot_2*.

As shown before, components can be configured using a *ComponentConfiguration*. Such a configuration is defined to be suitable for one or more components and for one or more Task(s) to be performed. For the task *calibrate_robot_1* only for the component *robot_pose_estimator_1* a configuration is modelled. This configuration of type *ComponentConfiguration* references an individual of type *ParameterConfiguration*, which configures the individual of type *ReconfigurableParameter* with name *activate* and type *Bool* to *true*. This means that the Master Control will at the end of the configuration process reconfigures each node for which a suitable configuration for the task is defined. For task *calibrate_robot_1* the parameter is used to activate the calibration of the robot, which causes the *robot_pose_estimator_1* to perform the configuration and to signalize the end of the calibration process on a topic.

The same is performed for task *calibrate_robot_2*, whereat *lwr_1* is replaced by *lwr_2* and *calibrate_robot_1* is replaced by *robot_pose_estimator_2*. The driver of *lwr_1* and *lwr_2* is the same component, which means that at the end the component list to use contains the same components as for the first calibration task, with the exception of the component *robot_pose_estimator_2*. However *robot_pose_estimator_2* uses the same executable as *robot_pose_estimator_1*, which is executed with a different name and is parametrized to calibrate *lwr_2*. This allows to use the same *ComponentConfiguration* as used for *robot_pose_estimator_1* in the previous task. The finish event in this scope signalizes the end of the calibration process for *lwr_2*.

5.3.2. Capturing of trocars

In the tasks *capture_trocar_1* and *capture_trocar_2* the position of the remote centers of motion for the two robots are captured. This is necessary as the position of the trocars has to be available to the telemanipulator as the shaft of the instruments attached to the robot and introduced via the abdominal wall are required to be aligned with the trocars at any time. This can be done using a pointing device that can be observed by the ARTTrack system. The position of the trocars in the robot base frames can be computed using the known position of the robot bases in the ARTTrack base frame as computed in the tasks *calibrate_robot_1* and *calibrate_robot_2*. For this purpose the components *robot_pose_estimator_1* and *robot_pose_estimator_2* keep running after the calibration process and will constantly output the position of the robot base frames in the ARTTrack base frame. The position of the trocars has to be signalized to the telemanipulation component that allows to control the robots via the input devices as this component computes the position of the robot's end-effector frame based on the desired position of the tool as commanded via the input devices and the position of the trocar for the

associated robot. The capturing of the trocar can be seen in Fig. 5.25. The telemanipulation component is modularized and contains a sub-component that computes the end-effector position from a given position of the input device and the position of the trocar. For each robot an individual telemanipulation component is spawned and parametrized for the associated robot and input device.

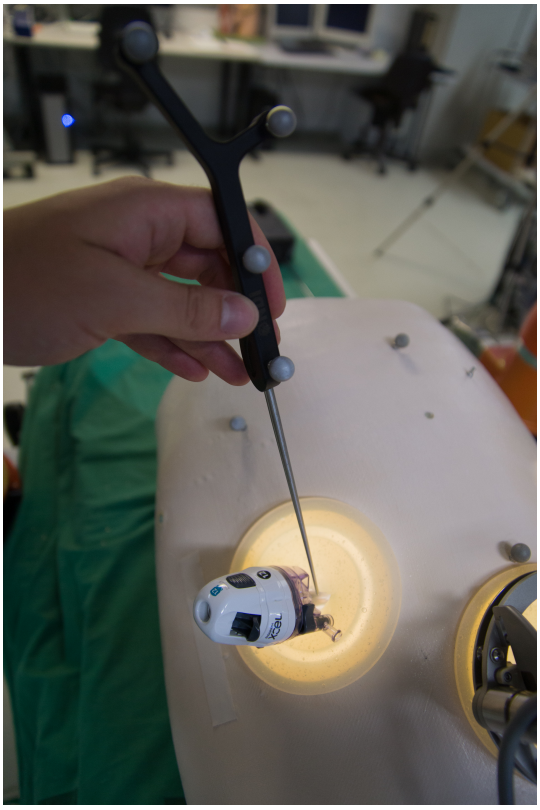


Figure 5.25.: The capturing process for one of the trocars

For both of the tasks, the Master Control checks if the component, which publishes the pointer tip with respect to the rigid body (known from pivoting), is available. It is also checked if the components to capture the trocar point and to pass it to the telemanipulator and the sub-component of the telemanipulator that computes the robot position from the trocar point, are *available*. Finally, as the personnel selects the trocar poses, a component has to be used, which allows to acknowledge that the current position of the pointer tip has to be set as trocar. This is done via a button on graphical user interface, which is also queried for availability by the Master Control.

During the capturing of trocars no configuration of components is required and the event that signalizes the end of the tasks is created by the component that captures the trocar point for the telemanipulator. In this scope a service is used, whereupon the service call will signalize the end of the task to the

Master Control. The service is called at the time where the trocars location is transmitted to the telemanipulator.

5.3.3. Robots move to start pose

In the tasks *robot_start_pose_1* and *robot_start_pose_2*, each of the robots is moved to a start pose. For this, the Master Control checks if the controller for each of the robots is running and if the component that commands the robot to a predefined starting pose is in place. If this is the case, the Master Control uses a *ComponentConfiguration* similar to the one used for the calibration tasks to activate the movement of the robot to the predefined start pose. The end of each task is signalized, when the robot is commanded to move to the start pose, using a service for each of the robots.

5.3.4. Install camera and tools

In the task *move_camera_start_pose* the tools are mounted and the endoscopic camera is introduced through a trocar using the Viky robot. the personnel acknowledges this via the graphical user interface that is also used for the capturing of the trocars. Therefore the Master Control has to query the graphical user interface and has to check for an event on a service that is called, when the user acknowledges the end of the installation process via a press on a button on the graphical user interface.

5.3.5. Installation finished

The task *camera_in_start_pose* is activated after the installation process and is the first task after which branches for the workflow exist. At this stage of the workflow the personnel can choose to use either the left hand or the right hand robot in hands-on mode. This is detected via the Kinect™ perception system. For this purpose the Master control has to check if the Kinect™ One system is running and responding and whether there are components in place that can interpret the output from the camera system. For the Kinect™ One system the Kinect™ 360 system is defined as an alternative. This means that the Master Control will first query the Kinect™ One system. If it fails, the Master Control will try the same for the Kinect™ 360 system.

The output data from each of the camera systems is of the same type and contains the same measures extracted from the scene. To check for the desire of an operator to use a robot in hands-on mode, the smallest distance between the end-effector of the robot and any human in the FoV of the camera system

is used, as the distance of the hands of the users to the robot is only robustly usable with the Kinect™One system. As hands-on detector, the approach of Schreiter et. al. [112] has been integrated. For the final experiments a moving average filter that filters the raw distances has been used. When the distance from the output of the filter was below a threshold, the detector signalized the switch to the hands-on mode. When it was above the threshold, the detector signalized the switch to the normal mode in which the robot can be controlled via a topic of the controller.

As the task has two successors, the hands-on detector has to run once for each of the robots. Therefore in the task, the Master Control has to check for the availability of both detectors as well as for the availability of the perception system. In this scope, the dynamic system composition is used as the hands-on detectors are subscribed to a topic on which they expect the data from the perception system to arrive. Depending on the camera system used, Kinect™ One or 360, the data is delivered by different components. This means that in the ontology a *ComponentConfiguration* is modelled, which in this case is suitable for both systems. As only one of the systems will be used, the output topics of the perception system are set to the input topics of the detection components. This shows how the system can adapt to available or unavailable components or failures.

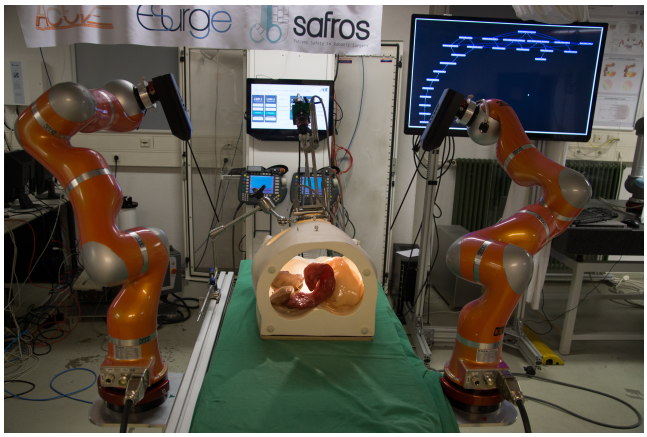


Figure 5.26.: Foreground: The phantom used for the evaluation and the devices in their starting pose prior to the telemanipulation process, Background right: Visualization of the workflow

For the task as there are two transitions to successors, two finish events are defined. The Master Control is therefore configured to listen for the outputs of the hands-on detection components using the knowledge from the ontology. If the hands-on detector for the left robot sends *"handson"* and therefore detects the desire to use the robot in hands-on mode, the task is finished and the task *hands_on_mode_robot_1* is activated. The same applies for the hands-on

detector for the right robot and task *hands_on_mode_robot_2*. Fig. 5.26 shows the robots and the endoscopic camera in their starting pose at the end of the task *camera_in_start_pose*.

5.3.6. Hands-on usage of the robots

The tasks *hands_on_mode_robot_1* and *hands_on_mode_robot_2* occur twice in the workflow. In these tasks the operator uses either the left or the right robot in hands-on mode as can be seen in Fig. 5.27

Figure 5.27.: An operator uses one of the robots in hands-on mode. In the top of the image parts of the perception system can be seen.

The first occurrence of the task is before the telemanipulation and allows to introduce the surgical instruments through the defined trocars. The second occurrence is after the telemanipulation task and allows to relocate the tools or to remove the tools from the patient's body. If one of the tasks is started, it is already known that the operator desires to use the robot in hands-on mode. This means that the Master Control checks for the presence of the robot to be used and the associated driver. For this driver a *ComponentConfiguration* (including a *ParameterConfiguration* of a *ReconfigurableParameter* of the robot

to change the control modes) has been defined that switches the robot from position control mode, where the driver accepts targets for the robot on a topic, to hands-on mode, where the operator can move the robot by applying forces.

Each of the tasks ends as soon as the associated hands-on detector signalizes that the operator released the end-effector of the robot via an according event on a topic. The Master Control has to configure a *"control loop"* to listen for the output of the detector that signalizes a switch to the position control mode. The Master Control will also check for the availability of the hands-on detector and a perception system for the task. For *hands_on_mode_robot_1* the only possible successor task is *fixed_mode_robot_1* and for *hands_on_mode_robot_2* the only possible successor task is *fixed_mode_robot_2*.

5.3.7. Robots in position control mode

In the tasks *fixed_mode_robot_1* and *fixed_mode_robot_2* the according robots are used in position control mode. Similar to the hands-on mode related tasks, the tasks *fixed_mode_robot_1* and *fixed_mode_robot_2* occur twice during the workflow. The purpose of the tasks is identical for both occurrences. However, the transitions differ. After the first occurrence, both tasks have the same successor tasks, which are *telemanipulation_bimanual*, *hands_on_mode_robot_1* and *hands_on_mode_robot_2*. This allows to reposition each robot as often as the personnel desires to relocate the robots before the telemanipulation task has begun and finally to switch to the telemanipulation task. In this scope, cycles are built within the workflow, as the task *hands_on_mode_robot_1* is defined as successor for both *fixed_mode_robot_1* and *fixed_mode_robot_2*, but is also a predecessor of *fixed_mode_robot_1*. The task *hands_on_mode_robot_2* is defined as successor for both *fixed_mode_robot_1* and *fixed_mode_robot_2*, but is also a predecessor of *fixed_mode_robot_2*.

At the start time of each of the tasks, it is known that the associated robot should be used in position control mode, where it is not moving as no data is sent to the robot controller. To initiate this, a *ComponentConfiguration* (including a *ParameterConfiguration* of a *ReconfigurableParameter* of the robot to change the control modes) is used that configures the robot controller to work in position control mode for the robot associated to the task.

In terms of outgoing transitions and required components, both tasks are similar to the *camera_in_start_pose* task due to the possible successors. Additionally, the Master Control has to listen for an event, which signalizes the start of the telemanipulation task (*telemanipulation_bimanual*). This event is signalized via a topic of the telemanipulator. This topic is published by the dead man switch that is used by the telemanipulator. This switch allows to

couple and decouple the input devices with the robots and sens a boolean *true* as soon as the pedal gets pressed.

The event therefore occurs, when a *true* from the dead-man switch is received, which causes the Master Control to finish the task and to initiate the transition to *telemanipulation_bimanual*. The other used events and transitions are taken from the task *camera_in_start_pose*.

As written before, the tasks *fixed_mode_robot_1* and *fixed_mode_robot_2* occur twice in the workflow. The second occurrence is located after the hands-on tasks that follow the telemanipulation task. In this case both tasks have only one transition. This transition for *fixed_mode_robot_1* leads to *trocar_inside_outside_detection_1* and for *fixed_mode_robot_2* leads to *trocar_inside_outside_detection_2*. This serves the purpose to check whether the tool is still inserted through the trocar, which means that the robot can be relocated in hands-on mode or the telemanipulation can continue, or if the tool has been extracted, what in the example workflow means that the robot will autonomously drive to a predefined finish pose.

Both tasks are implemented once in the ontology with all four possible transitions. During the execution, the parsed graph from the workflow file is used to check, which transitions are actually possible in the current state of the procedure and which transitions have to be neglected.

The event to signalize the transition from tasks *fixed_mode_robot_1* and *fixed_mode_robot_2* to their successors *trocar_inside_outside_detection_1* and *trocar_inside_outside_detection_2* is also initiated via the event that signalizes a change from hands-on mode to position control mode. This means that the Master Control compares the output of the associated hands-on detector with the reference for the end of the hands-on mode. This assures that the human that moves the robot in the prior hands-on step is not anymore in proximity to the end-effector. This is necessary as the robot will autonomously move to a finish pose after this check in case the tool has been extracted from the patient's body.

5.3.8. Telemanipulation

In the task *telemanipulation_bimanual*, the actual procedure is performed. The surgeon is situated at the master console, which can be seen in Fig. 5.28 and controls the robot via two haptic input devices by ForceDimension (Nyon, Switzerland). A dead-man switch is used, which has to be pressed during the operation. The surgeon's view is realized via an endoscopic camera, whose image is displayed on a monitor at the master console. The camera

is registered to the optical tracking system using a rigid body and a checkerboard based registration method. The pose of the tool tip of the instruments, attached to the end-effector, with respect to the robot base frame is known from the CAD file for the instrument and the forward kinematics of the tool. For this work, the tool as shown in the work of Hutzl et al. [57] is used, which is equipped with a Motor for the opening and closing of the gripper/scissors and a motor to rotate the tool around the shaft of the instrument. Using this tool and the remote center of motion constraint, the surgeon can modify the position of the tool tip, can rotate the tool around its shaft and can open and close the gripper.

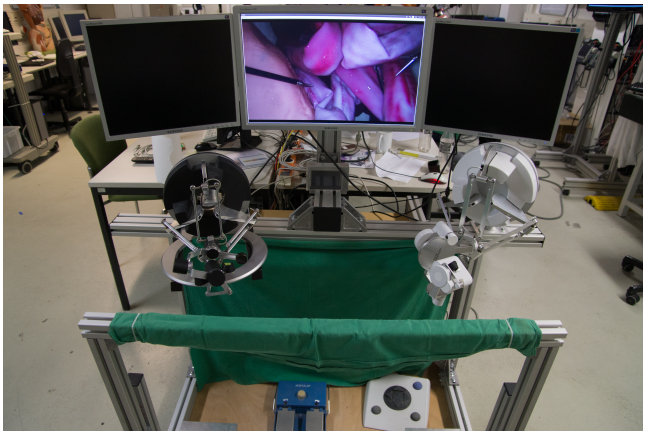


Figure 5.28.: The master console for the telemanipulator

As the trocar-, the robot base-, the camera coordinate- and the tool-tip-frame are all known in the base frame of the optical tracking system, the tool tip is moved in the coordinate frame of the endoscopic image, which allows for a natural interaction with the system and for the compensation of the Fulcrum-effect that is introduced by the trocar and which is depicted in Fig. 5.29. A simplified transformation chain for the telemanipulation system is shown in Fig. 5.30.

At the time the dead-man switch is pressed, the current poses of the camera, the input devices and the tool-tips are captured and the surgeon can move the robot tool-tip relatively to these poses in the camera coordinate frame. If the surgeon desires to reposition the camera, or if he/she reaches the end of the workspace of one of the input devices, he/she can release the dead-man switch, then reposition the input devices/camera and then press and hold the dead-man switch again, which will capture the new positions and causes the telemanipulator to perform the motions, relatively to the newly captured positions. The slave side of the telemanipulator as used during the procedure can be seen in Fig. 5.31. The screenshot from the video captured by the endoscopic camera during phantom experiments can be seen in Fig. 5.32.

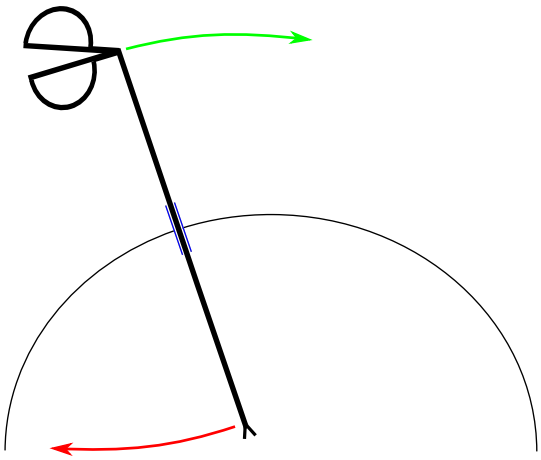


Figure 5.29.: Schematic description of the Fulcrum effect when a trocar/remote center of motion (blue) is used. The movement of the handle (green) results in a mirrored movement of the tool tip (red)

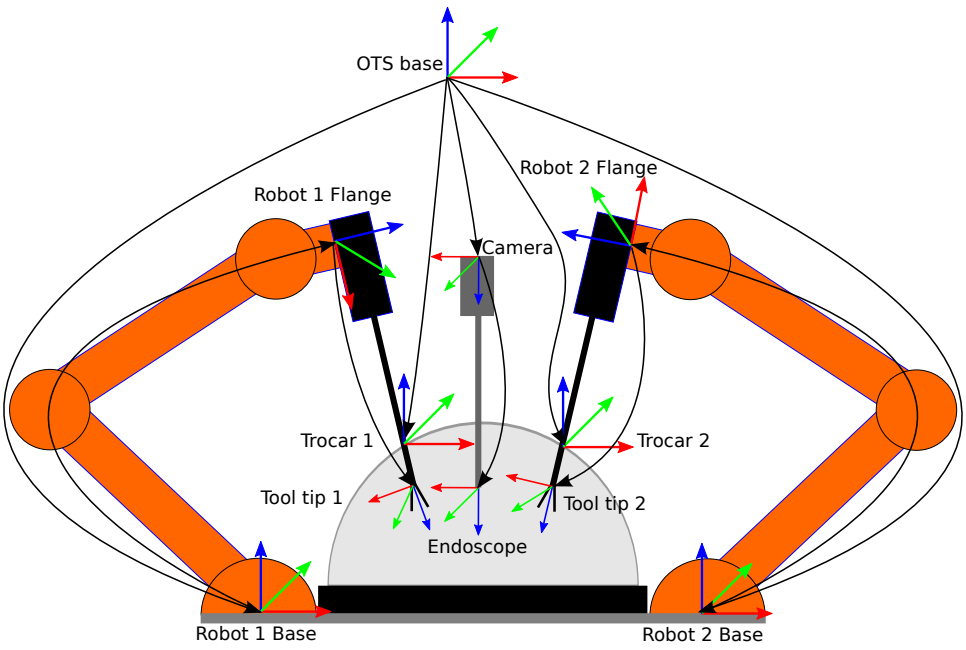


Figure 5.30.: Schematic and simplified representation of the coordinate frames in the telemanipulation system.

The task *telemanipulation_bimanual* ends as soon as an operator desires to move one of the robots in hands-on mode. This means that the Master Control has to utilize the hands-on detectors, as well as a perception system as input for the hands-on detectors in this task. The actions to be taken as implemented in the knowledge base are identical to the ones used in the *camera_in_start_pose*. Additionally, the Master Control checks the telemanipu-

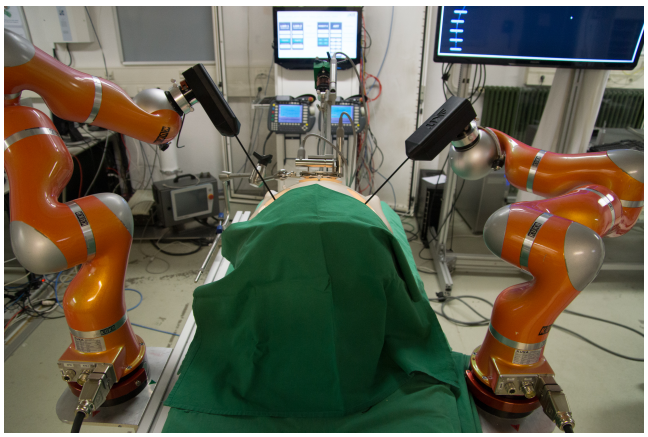


Figure 5.31.: The slave side of the telemanipulator during phantom experiments

Figure 5.32.: Screenshot of the video as captured by the endoscopic camera in the robotic configuration as shown in Fig.5.31

lation components for availability prior to the execution of the task, to assure that the telemanipulator is functional.

The task is succeeded by either task *hands_on_mode_robot_1* or task *hands_on_mode_robot_2*.

5.3.9. Check if tool is inserted through the Trocar

In the tasks *trocar_inside_outside_detection_1* and *trocar_inside_outside_detection_2*, the system utilizes the telemanipulator to check whether each of the tools is inserted through a trocar. For this purpose, the component checks if the z-component of the tool tip position in the optical tracking system frame is smaller than the z-component of the trocar position. The z-axis of the optical tracking system frame is aligned with the z-axis of the floor, which means

that the floor has a z-value of 0.0 m and the ceiling has a z-value equal to the ceiling height.

Each of the tasks has three possible transitions, which lead to the telemanipulation task, the hands-on task of the robot or to the task that moves the robot to the predefined finish position. To execute the Task, the Master Control checks if the telemanipulator components are *available* that can check for the position of the tool-tip with respect to the trocar. It is also checked if the hands-on detector for the associated robot is *available* and if a suitable perception system is running to provide the data to the hands-on detector.

If the tool tip is extracted from the patient's body, the telemanipulator will signalize this using a boolean *true* on a topic, which the Master Control checks for, as an event for initiation of the transition to the task that moves the associated robot to its finish pose. If this is not the case, the Master Control will check for the hands-on event that is signalized via the hands-on detector and for the dead-man switch pressed event that is signalized via the telemanipulator.

5.3.10. Move robots to finish pose

The tasks *robot_finish_pose_1* and *robot_finish_pose_2* serve the purpose to transfer the robots to a final pose, where the tools can be dismounted and the robots can be removed from the surgical bed. The poses for the robots equal the start poses and therefor the same components as used for *robot_start_pose_1* and *robot_start_pose_2* are used. This means that also the same components as for these tasks have to been checked for availability and that the same individuals of type *ComponentConfiguration* can be used to initiate the robot movements. Both tasks finish as soon as the robots are commanded to their finish pose. The successor of *robot_finish_pose_1* is *trocar_inside_outside_detection_2* and the successor of *robot_finish_pose_2* is the output condition of the process, which finishes the workflow. This means that in the current implementation of the workflow, for simplification of the process, the left hand tool has to be extracted first.

6. Results

The concepts and algorithms shown in chapters 4 and 5 are evaluated in this chapter. Both the perception system and the workflow-based controller have been evaluated in separate sections. The camera system has been evaluated with respect to accuracy, performance and error rate. For the workflow-based controller the workflow described within chapter 5.2.2 has been used. Test person were instructed to execute the workflow on the system and their experience has been measured with questionnaires. Additionally, the performance and failure rates of the workflow-based controller have been measured. The chapters are divided into the sections experimental protocol, where the protocols used for the evaluation are described and experimental results, where the results are presented.

6.1. Perception system

In order to evaluate the performance of the camera systems, both registration/calibration accuracies and the actual system performance have been measured and evaluated. In order to use the system, a precise registration and calibration process is required. To evaluate the algorithms, the accuracy of the calibration and registration algorithms is measured.

6.1.1. Experimental protocol

6.1.1.1. Registration accuracy

For the Kinect™ 360 system both, the pairwise registration algorithm, as described in chapter 5.1.1.1, and the registration algorithm to calibrate with respect to an external tracking system, as can be seen in chapter 5.1.1.2, have been evaluated. For the Kinect™ One system, the registration algorithm to calibrate with respect to an external tracking system has been evaluated. For the evaluation the camera systems have been registered using the proposed algorithms. Then the registration error between each camera and the reference is evaluated by calculating the locations of known objects in the local coordinate frames of both, the camera and the reference system. Using

the transformation from the registration, the locations of the objects can be transformed to the reference coordinate frame. The registration error can then be computed by calculating the euclidean distances between the object's positions as perceived by the referencing system and the camera that has been registered to the reference.

For the evaluation of both algorithm the checkerboard was used as known object and the location of the checkerboard corners has been measured and transformed to the coordinate frame of the referencing system as stated above. Depending on the algorithm used, either the ARTTrack2 system or one of the cameras has been used as reference.

6.1.1.1.1. Pairwise registration algorithm For the pairwise registration algorithm, the checkerboard locations have been collected using both the RGB sensor as well as the depth sensor of the Kinect™ 360. Therefore it is assumed that the factory calibration between the RGB sensor and the depth sensor is optimal. This means that for each pixel of the RGB sensor, there exists a corresponding location on the depth sensor. The checkerboard corner locations are computed using OpenCV and the corresponding depth value is used to estimate the 3D location of every corner of the checkerboard in the field of view of the Kinect™ 360 cameras. For this process, the checkerboard is put to a stable position in both the FoV of the reference camera and the FoV of the camera that is to be registered.

A checkerboard of size 6x5 with a field size of 0.08 m x 0.08 m has been used both for calibration and registration. This results in a maximum of 30 corresponding locations of checkerboard corners for both cameras. Even though the depth values for every location of a checkerboard have been collected over 60 frames, it is possible that none of these frames contains a valid depth location for one or more checkerboard locations, due to incomplete depth information acquired by the Kinect™ 360 camera. If one or more valid depth values for a checkerboard corner are available, the depth is averaged over the number of depth values and is used for the computation of the position in 3D space. This is repeated for all checkerboard corners in both camera frames. As shown in chapter 5.1.1.1 it is always known, which checkerboard corners in multiple camera frames correspond with each other. For each corner of the checkerboard, corresponding locations are collected. For the registration as well as the evaluation only correspondences are used, where valid depth locations for the checkerboard corner exist in both camera frames. Other correspondences are neglected. This results in typically 20-25 valid correspondences for a checkerboard location instead of a theoretical maximum number of 30.

For the evaluation, the checkerboard has been placed at 12 locations within the

FoV of the cameras after they have been registered. The evaluated workspace was of size 1.8 m x 1.0 m x 1.5 m. This resulted in 12 sets of correspondences. Using the known transformation from the registration, the locations of the corners within the correspondences have been transformed to the reference frame and the euclidean distances between them have been computed and represent the errors. The data has then been statistically analysed using a Kruskal-Wallis method. The measured error gives an estimate about how well the point clouds of the camera system, registered with the pairwise registration algorithm, are aligned to each other. This experiment is referred to as *experiment 1* in the following.

6.1.1.1.2. Registration to external reference For the registration using the algorithm to register a camera to the external referencing system, no depth information has been used and only the IR camera has been used as 2D source for the detection of known objects. When the algorithm is used with the Kinect^{TM} 360 system, the RGB camera is utilized instead of the IR camera and the factory registration between the IR and the RGB camera is used. The checkerboard used for the calibration with this algorithm was of size 4x3 with a field size of 0.096 m x 0.096 m. In contrast to the processing of depth information, as used in the pairwise registration algorithm, the estimation of the checkerboard pose within the 2D image using the Levenberg-Marquardt optimization allows to compute 3D values for each of the corners of the checkerboard within the field of view and therefore no invalid depth information can occur. This means that a complete set of 12 checkerboard corner locations for each checkerboard position in the camera frame was available for the registration process, from which the four outer corners have been used as shown in chapter 5.1.1.2.

In order to evaluate both algorithms against each other, the algorithm to register a camera against an external reference has been analysed using the evaluation process described in chapter 6.1.1.1.1. As the algorithm in contrast to the pairwise registration algorithm does not provide the transformation between two cameras, this transformation had to be computed to allow for the evaluation process. Therefore the camera used as reference for the pairwise registration algorithm has been registered to the optical tracking system. The same is repeated for all other cameras. As all cameras are registered with respect to the optical tracking system, the transformations $K_{cam2,cam1}$, $K_{cam3,cam1}$ and $K_{cam4,cam1}$ from each camera to the camera used as a reference for the pairwise registration algorithm can be computed. The computed transformations have then been used for the evaluation process described in chapter 6.1.1.1.1, whereas the same 6x5 sized checkerboard has been used. Again 12 checkerboard locations have been used, which were distributed over an workspace of size 1.8 m x 1.5 m x 1.0 m and the data was statistically

analysed using a Kruskal-Wallis method. The measured error gives an estimate about how well the point clouds of the camera system, registered with the algorithm to register a camera against an external reference, are aligned to each other. This experiment is referred to a *experiment 2* in the following.

In order to evaluate the registration accuracy of both camera systems against each other, two additional experiments have been set up that are based on the algorithm to register a camera against an external reference. In *experiment 3*, the Kinect™ 360 system has been registered and evaluated using the algorithm to register a camera to an external reference. In *experiment 4*, the same has been performed for the Kinect™ One system. The evaluation process is identical for experiments 3 and 4 and is described in the following.

As shown in chapter 5.1.1.2, the optical tracking system used cannot detect the corners of a checkerboard without manual interaction. Therefore, the referencing system is not directly able to collect a point cloud from a reference object. Both experiments therefore do not measure the alignment of point clouds, but the registration error between each camera and the optical tracking system.

For each camera to be evaluated, the checkerboard has been placed at 12 locations within the FoV of the optical tracking system and the camera that has been registered with respect to the optical tracking system. For each checkerboard pose, the IR sensor of the Kinect™ has been used to find the pose of the checkerboard using its known geometry and to estimate the locations of the four outer checkerboard corners in the local coordinate frame of the IR sensor. Then each outer corner location is collected in the frame of the optical tracking system using a pointing device. The checkerboard corner locations in the Kinect™ cameras frame have then been transformed to the coordinate frame of the optical tracking system using the transformation that has been computed using the algorithm to register a camera to an external tracking system. The euclidean distance between corresponding checkerboard corner locations, collected using the Kinect™ and the optical tracking system, has been used as measure for the registration error.

The evaluated workspace was of size 1.0 m x 1.0 m x 1.0 m. For each checkerboard corner location, correspondences in both the optical tracking coordinate frame and the camera coordinate frame were collected. For all 12 checkerboard locations and the four outer checkerboard corners, this resulted in 48 checkerboard corners that have been used for the evaluation process. The data has then been statistically analysed using a Kruskal-Wallis method. This gives the registration accuracy between the optical tracking system and each camera that is registered with the optical tracking system.

6.1.1.2. Robot localization accuracy

The perception system as shown in chapter 5.1 is able to compute distances between objects and humans. In order to give an estimate about the overall performance of the system, all registration errors have to be considered. The distance computation between objects and humans is influenced not only by the camera errors and the registration errors but also by the localization error of objects that are used to compute distances in between. The primarily used robot in this work is the KUKA LWR4. As can be seen in chapter 5.1.1.4, the optical tracking system is used to localize the base pose of the robot. In this chapter, the resulting position error of the robot's flange with respect to the optical tracking system is evaluated, which originates from the base localization and the robot's positioning error. For this purpose, the robot is localized using the algorithm from chapter 5.1.1.4. Then the robot is randomly moved to 400 poses in a workspace of size 0.1 m x 0.1 m x 0.1 m while the orientation is kept steady. For each of these poses, the robot flange pose is measured using the robot's kinematics and the optical tracking system. The distance of the two measured flange poses for a pose gives the localization error of the robot's flange with respect to the optical tracking system. In order to compute this error the following transformation chain is used:

$$E = K_{Flange,Body} * K^{-1}_{OTS,Body} * K_{OTS,Base} * K_{Base,Flange} \tag{6.1}$$

In this scope, $K_{OTS,Body}$ is the pose of the coordinate frame of the markers, attached to the robot, within the coordinate frame of the optical tracking system. $K_{Base,Flange}$ is the transformation from the robot's base to the robot's flange as measured using the robot's kinematics. $K_{OTS,Base}$ is the transformation from the optical tracking system to the robot's base as computed using the robot localization algorithm. $K_{Flange,Body}$ is the fix transformation from the rigid body to the robot's flange, which is estimated using pivoting of the rigid body, attached to the robot flange, around the flange.

The result from this transformation chain is the position error of the robot flange with respect to the optical tracking system. From this error, the euclidean distance between the robot's flange in the base frame and the optical tracking system frame is computed and used for evaluation purposes. This euclidean distance is captured at each of the 400 poses.

6.1.1.3. Temporal behavior, noise and tracking accuracy

In preliminary experiments with the Kinect™ One system, a small time delay of the output data from the perception system has been observed. The temporal behavior was evaluated for the Kinect™ One system only, as for

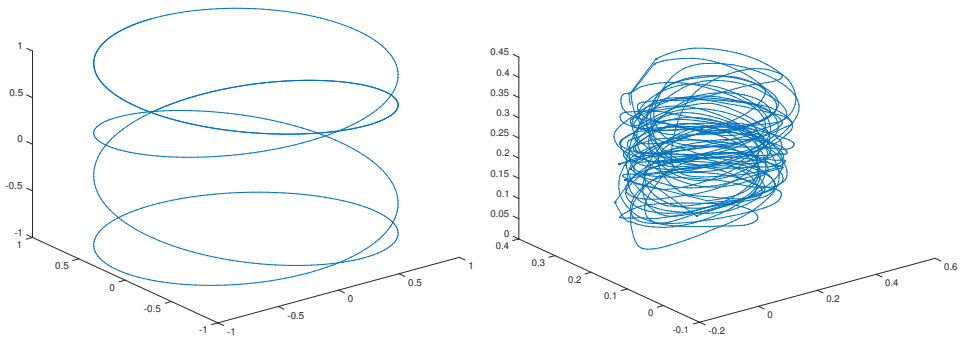


Figure 6.1.: Left: Lissajous shaped movement to be performed. Right: Actual performed hand movement

the previously developed Kinect™ 360 system the data was available in the literature. The delay with respect to the action in the physical perceived scene has been evaluated using the protocol shown below.

The data from the ARTTrack2 (OTS) has been used as ground truth. The time delay of this system is given as <20 ms. A marker-equipped rigid body has been attached to the back of a hand of a human that served as test person. The optical tracking system was then used to capture the location of the rigid body at any time. Simultaneously, the perception system's output data was collected. Both, the delay of the full body tracking from the fused camera data and the delay of the distance computation have been evaluated. Two experiments have been set up for this purpose

In the first experiment, the geometry of robot and test person was set up so that the closest distance in between is always the distance between the hand and the flange of the robot. The test person performed periodic hand movements with a frequency of f<1 Hz. As extreme values, distance of 0.0 m and 0.5 m between the hand and the flange of the robot have been used during the movement. The sampling frequency was 30 Hz and 2000 samples have been collected from the ground truth and from the perception system. The location of the flange of the robot within the coordinate frame of the ground truth was known and was used to compute the distance of the rigid body from the flange. Both vectors of distance have been treated as a periodic signal and cross correlation has been used to identify the time delay in between the signals.

For the time delay of the fused skeleton tracking, the ARTTrack2 was used as ground truth. A rigid body was attached to the test person's right hand. A 3D Lissajous-shaped hand movement as can be seen in Fig. 6.1 in the FoV of two of the Kinect™ One cameras and the ARTTrack2 has been performed.

3000 samples have been collected with a sampling frequency of 30 Hz. The locations of the hand have been captured using each of the Kinect™ One

cameras, the fusion algorithm and the ARTTrack2 system, which results in 4 sets of hand locations. As the signal was periodic on every axis, movements on one axis on its own were suitable for the time delay evaluation. The hand locations on the x-axis captured using both the ground truth and the fusion algorithm have been treated as signals to compare. Cross correlation has been used to identify the time delay between the signals.

The data from the second experiment has also been used for the evaluation of the noise level of the different tracking modalities and algorithms. For this purpose, the signals have been aligned using the analyzed time delay. For each tracking data from the Kinect™ One cameras and for the tracking data from the fusion algorithm, the euclidean distance to the ground truth (ARTTrack2) data has been computed to get an estimate about the registration error of the depth data from the Kinect™ One system and to get an estimate about tracking error of the tracking methods.

Finally the data has also been used to compare the noise, produced by each of the tracking systems and methods, to each other. For this purpose the noise has been extracted from the signal using a equiripple FIR high pass filter with a stopband frequency of 3.5 Hz, a passband frequency of 4 Hz, a passband ripple of 1 dB and a stopband attenuation of 60 dB The fast fourier transform has been used to determine the noise characteristics.

6.1.1.4. Kinect™ One interferences

During the experiments with the Kinect™ One camera several objects changed locations by a few cm with a high frequency. This behavior was occasionally observable but seemed to have a periodic behavior and only occurred when more than one Kinect™ One was active. Therefore the assumption was made that the Kinect™ cameras interfere with each other and the radiation of two cameras has been measured using photodiodes and an oscilloscope. Using the oscilloscope, a 30 Hz carrier signal of the active illumination was measured. The frequency of the two cameras' signals were slightly different and the signals dephased and rephased periodically. The interferences were present, when the signals were dephased. Ways to reduce the observed interferences were not assessed, as the effect of the interferences was relatively small and did not affect the system to a high extent with respect to the target application.

6.1.1.5. Performance evaluation

In order to evaluate the overall system performance, the tracking and detection performance of the systems have been evaluated. Two test persons

have been asked to perform the experiments, whereas test person 1 was a 1.93 m tall male and test person 2 was a 1.63 m tall female. Test persons with different height and figure have been chosen to evaluate how the systems are affected by these variables. The output data of the Kinect™ One system was found to be more stable compared to the Kinect™ 360 system. The less stable Kinect™ 360 output data made an automatic evaluation of the raw output data hard and therefore all experiments have been observed by a human operator, who annotated the data using a stopwatch. The output data of the Kinect™ One system was found to be more stable and the output data has been collected and analyzed using MATLAB (The MathWorks Inc., Natick, MA, USA). The test protocols used for both camera systems are shown in the following:

6.1.1.5.1. Kinect™ 360 Five tests have been set up for the Kinect™ 360 system and the test persons were wearing blue operating room clothing:

1. **Slow movement test:** The test person was moving in the FoV of the camera system with a speed of 0.0-0.5 m/s. The human operator observed both the real scene and the virtual representation. The time is measured until the test person is lost by the system. The test has been performed for a duration of 3 minutes and has been repeated 3 times.

2. **No movement test:** The test person completely stops its movement and the time until the system loses the test person is measured. 13 iterations have been performed.

3. **Hand movement test:** The test person is standing close to the operating table and performs hand movements mimicking surgical actions. The time until the system loses detection of the test person was measured.

4. **Detection time test:** The test person starts moving. The time until the person is detected by the system is measured. 13 iterations have been performed.

5. **False-positive-classification test:** A second person enters the FoV of the camera system and approaches the test person. As soon as the system falsely classifies both humans as one human the distance is measured. 13 iterations have been performed. The test determines the distance range for false-positive classification results of the Fusion algorithm, which means that two persons are falsely detected as a single person.

Full body tracking with the Kinect™ 360 system was found to be not stable and tests have been performed with the final Kinect™ One system only.

6.1.1.5.2. Kinect™ One The test cases used in the experiments for the Kinect™ 360 system have also been used for the Kinect™ One system. Due to the robustness of the Kinect™ One system the no movement, slow movement and hand movement tests have been combined for the Kinect™ One system.

1. **Detection time test:** The test person enters the FoV of the camera system. The time is measured until the camera system gets track. The time is computed using the output data of the system from which the user count can be computed. A countdown was provided to the test persons after which they started to move into the FoV. The limit of the FoV was marked on the floor at a distance of 3.4 m from the camera rig in order to mark the starting position for the test person. The time from the end of the countdown until the user count increased by 1 is the detection time.

2. **Performance test:** The test person moves in the FoV of the camera system with a speed of 0.0-0.5 m/s. The test case includes periods where no movement and only hand movement are present. The test has been performed for a duration of 3 minutes and has been repeated 3 times. Each repetition includes a phase of 30 seconds duration with no movements and a phase of 30 seconds duration with hand movements only. During the *hands movements only* phase the test person used laparoscopic surgical tools and mimicked motions similar to movements during surgical procedures. The system output is used to verify how often and how long the system failed to detect and to track the human.

3. **False-positive-classification test:** This test is identical to the false-classification test used for the Kinect™ 360 system.

6.1.2. Experimental results

In this section the experimental results of the proposed experimental protocols for the perception system are shown

6.1.2.1. Pairwise registration algorithm

6.1.2.1.1. Experiment 1 The approach shown in chapter 6.1.1.1.1 has been applied to the Kinect™ 360 system. Every Kinect™ 360 K_2 to K_4 has been registered with respect to K_1. For camera pair 1 (K_1, K_2) 291 correspondences, for camera pair 2 (K_1, K_3) 284 correspondences and for camera pair 3 (K_1, K_4) 250 correspondences were collected for the 12 poses of the checkerboard.

Table 6.1.: Median values and quartile range in mm for the point cloud registration error evaluation of each group of Kinect™ cameras in experiment 1.

Pair	Median Value	1st quartile	3rd quartile
Pair 1	19.84	12.74	28.21
Pair 2	26.89	19.90	34.80
Pair 3	26.68	20.24	35.14

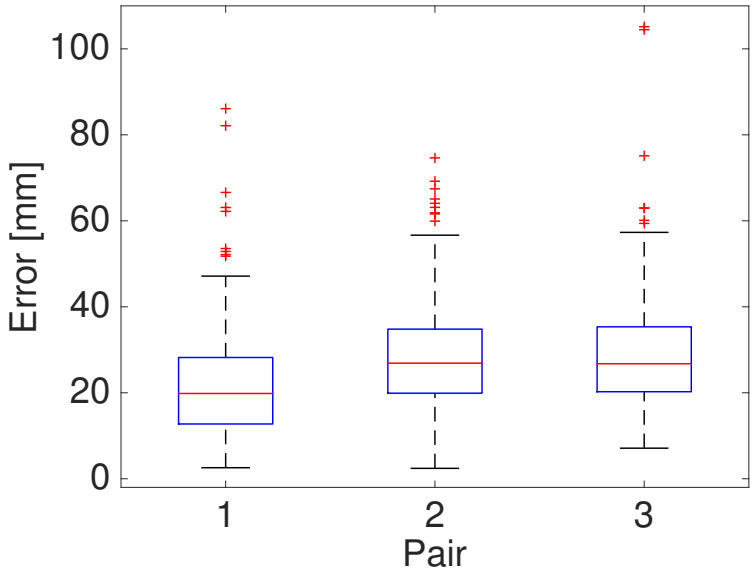


Figure 6.2.: Boxplot showing the point cloud registration error in mm for each Kinect™ 360 cameras, registered to the reference camera using the pairwise registration algorithm

The median values as well as the inter-quartile ranges for the errors from these data sets are shown in table 6.1. In Fig. 6.2 the results of the accuracy evaluation of the pairwise registration algorithm are shown. The detailed registration error is shown individually for each checkerboard position and each pair of Kinect™ 360 in Fig. A.1 (Annex 1). The Kruskal-Wallis analysis of the data lead to p-values of $p<0.005$ for the comparison of pair 1 and 2, $p<0.005$ for pair 1 and 3 and $p=0.7065$ for pair 2 and 3. No statistical difference between the point cloud registration errors of the pairs of cameras was used as Nullhypothesis. Therefore a high evidence for a significant statistical difference in between pair 1 and 3 as well as pair 1 and 2 is present. There is very low evidence for a statistical difference in between pair 2 and 3.

6.1.2.2. Registration to external reference

6.1.2.2.1. Experiment 2 The approach shown in chapter 6.1.1.1.2 has been applied to the Kinect™ 360 system and the results of *experiment 2* are shown below. For every Kinect™ 360, K_2 to K_4, the transformation to K_1 has been computed as K_1 has been used as reference in *experiment 1*. For pair 1 of cameras (K_1, K_2) a number of 186 correspondences, for pair 2 of cameras (K_1, K_3) a number of 161 correspondences and for pair 3 of cameras (K_1, K_4) a number of 130 correspondences were collected for the 12 poses of the checkerboard. The median values as well as the inter-quartile ranges for the errors from these data sets are shown in table 6.2

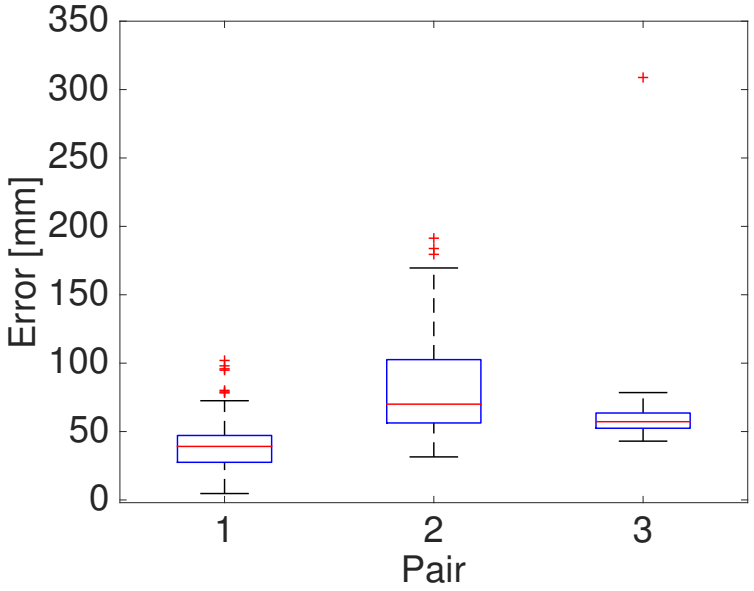


Figure 6.3.: Boxplot showing the point cloud registration error in mm for each Kinect™ 360 cameras, registered to the reference camera using the algorithm to register a camera to an external reference

Table 6.2.: Median values and quartile range in mm for the point cloud registration error evaluation of each group of Kinect™ cameras in experiment 2.

Pair	Median Value	1st quartile	3rd quartile
Pair 1	39.13	27.51	47.13
Pair 2	70.03	56.28	102.58
Pair 3	57.20	52.42	63.55

In Fig. 6.3 the results of the accuracy evaluation of the pairwise registration algorithm are shown. The detailed registration error is shown individually for each checkerboard position and each pair of Kinect™ 360 in Fig. A.2 (Annex

1). The Kruskal-Wallis analysis of the data lead to p-values of $p<0.005$ for the comparison of pair 1 and 2, $p<0.005$ for pair 1 and 3 and $p<0.005$ for pair 2 and 3. No statistical difference between the point cloud registration errors of the pairs of cameras was used as Nullhypothesis. Therefore a high evidence for a significant statistical difference in between pair 1 and 3 as well, pair 1 and 2 as well as pair 2 and 3 is present.

6.1.2.2.2. Experiment 3 Every Kinect™ 360 K_1 to K_4 has been registered with respect to the optical tracking system. For each Kinect™ 360 camera the checkerboard has been placed at 12 locations and the outer corners of the checkerboard were captured what lead to 48 checkerboard corner locations that were used for the evaluation. The median values as well as the inter-quartile ranges for the errors from these data sets are shown in table 6.3

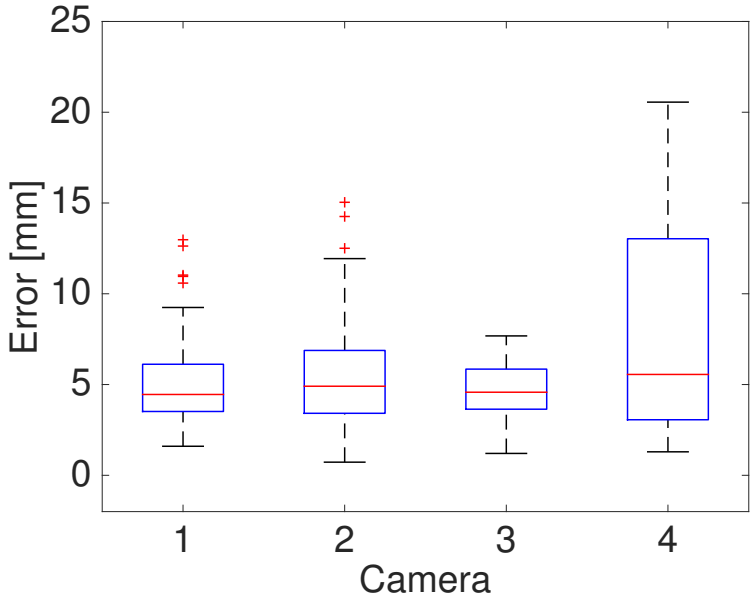


Figure 6.4.: Boxplot showing the registration error in mm for each Kinect™ 360 cameras, registered to the optical tracking system.

Table 6.3.: Median values and quartile range in mm for the registration error evaluation of each group of Kinect™ 360 cameras in experiment 3.

Camera	Median Value	1st quartile	3rd quartile
Camera 1	4.46	3.52	6.12
Camera 2	4.91	3.42	6.87
Camera 3	4.58	3.64	5.85
Camera 4	5.56	3.06	13.03

In Fig. 6.4, the results of the registration evaluation for experiment 3 are

shown. The detailed registration error is shown individually for each checkerboard position and each pair of Kinect™ 360 in Fig. A.3 (Annex 1). The Kruskal-Wallis analysis of the data lead to p-values of p>=0.2654 when the registration error distribution of all cameras is compared. No statistical difference between the camera registration errors was used as Nullhypothesis. Therefore there is very low evidence for a significant statistical difference in between the camera registration errors.

6.1.2.2.3. Experiment 4 Every Kinect™ One K_1 to K_4 has been registered with respect to the optical tracking system.
For each Kinect™ One camera the checkerboard has been placed at 12 locations and the outer corners of the checkerboard were captured, which lead to 48 checkerboard corner locations that were used for the evaluation. The median values as well as the inter-quartile ranges for the errors from these data sets are shown in table 6.4

Table 6.4.: Median values and quartile range in mm for the registration error evaluation of each group of Kinect™ One cameras in experiment 4.

Camera	Median Value	1st quartile	3rd quartile
Camera 1	9.14	5.83	13.60
Camera 2	10.28	6.66	14.04
Camera 3	7.55	5.59	8.66
Camera 4	5.56	3.92	7.93

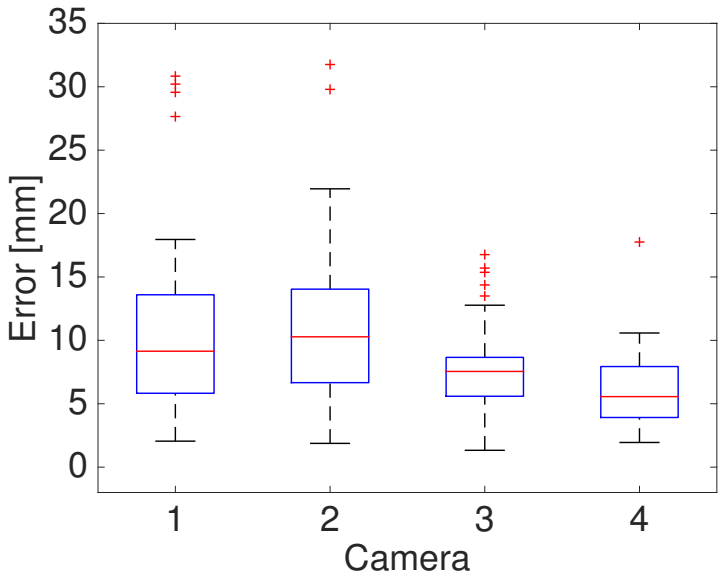


Figure 6.5.: Boxplot showing the registration error in mm for each Kinect™ One cameras, registered to the optical tracking system.

Table 6.5.: p-values from the Kruskal-Wallis analysis of experiment 4, where the registration error of each Kinect™ One camera is measured. The cells show the p-values from the pairwise comparison of the errors of the cameras in the pairs (column,row).

*	Camera 1	Camera 2	Camera 3	Camera 4
Camera 1	-	p=0.3298	p=0.0812	p<0.005
Camera 2	p=0.3298	-	p<0.005	p<0.005
Camera 3	p=0.0812	p<0.005	-	p=0.0089
Camera 4	p<0.005	p<0.005	p=0.0089	-

In Fig. 6.5, the results of the registration evaluation in experiment 4 are shown. The detailed registration error is shown individually for each checkerboard position and each pair of Kinect™ One in Fig. A.4 (Annex 1). The p-values form the Kruskal-Wallis analysis of the data are shown in table 6.5 No statistical difference between the camera registration errors was used as Nullhypothesis. As can be seen from the p-values for some combinations a high evidence for a significant statistical difference is present.

6.1.2.3. Robot localization accuracy

The localization error of the robot's flange has been measured as shown in chapter 6.1.1.2. The results of the evaluation are shown in table 6.6 and Fig. 6.6, whereas an outlier of 51.11 mm is not shown in the figure.

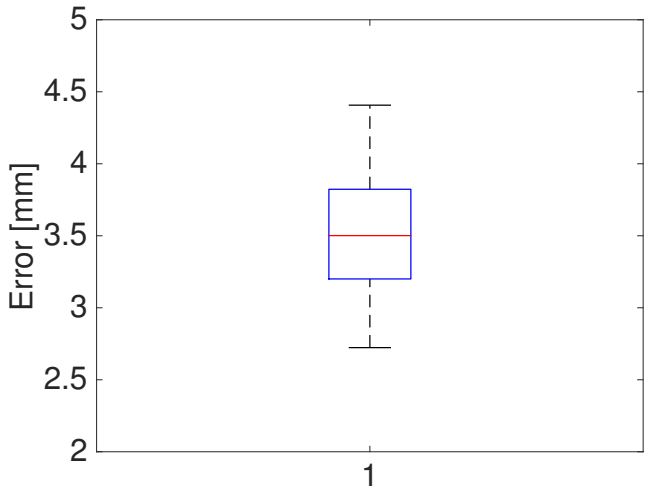


Figure 6.6.: Boxplot showing the registration error in mm for the robot localization.

Table 6.6.: Robot flange localization error in mm

Median Value	1st quartile	3rd quartile
3.50	3.20	3.82

6.1.2.4. Temporal behavior, noise and tracking accuracy

The following paragraph shows the results from the experimental protocols shown in chapter 6.1.1.3. The distance computation algorithm has been found to be 0.267 sec delayed with respect to the ground truth. The skeleton tracking fusion algorithm has been found to be 0.068 sec delayed with respect to the ground truth.

The combined tracking and registration error of the tracking data from each camera and of the fusion algorithm's tracking data is shown in Fig. 6.7 and table 6.7.

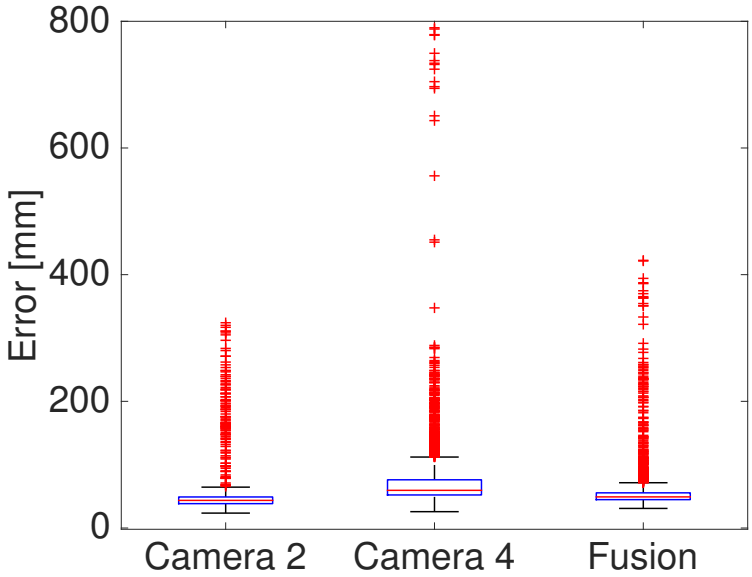


Figure 6.7.: Boxplot showing the combined tracking and registration error between the tracking algorithms and the ground truth.

The noise characteristics of the different tracking algorithms and data sources is shown in the following. The results are only shown for the noise of the signals on the x-axis of the ARTTrack2 tracking system. The noise characteristics of the signal's y- and z-axis components is similar but the amplitude was up to 4 times lower on the y-axis and up to 3 times higher on the z-axis.

The single-sided amplitude spectra of the noise are shown for each tracking method individually in Fig. 6.8

Table 6.7.: Median values and quartile range in mm for the combined tracking and registration error between the tracking algorithms and the ground truth (ARTTrack2).

Camera	Median Value	1st quartile	3rd quartile
Camera 2	43.60	38.49	49.03
Camera 4	59.48	52.19	76.14
Fusion	49.13	44.80	55.53

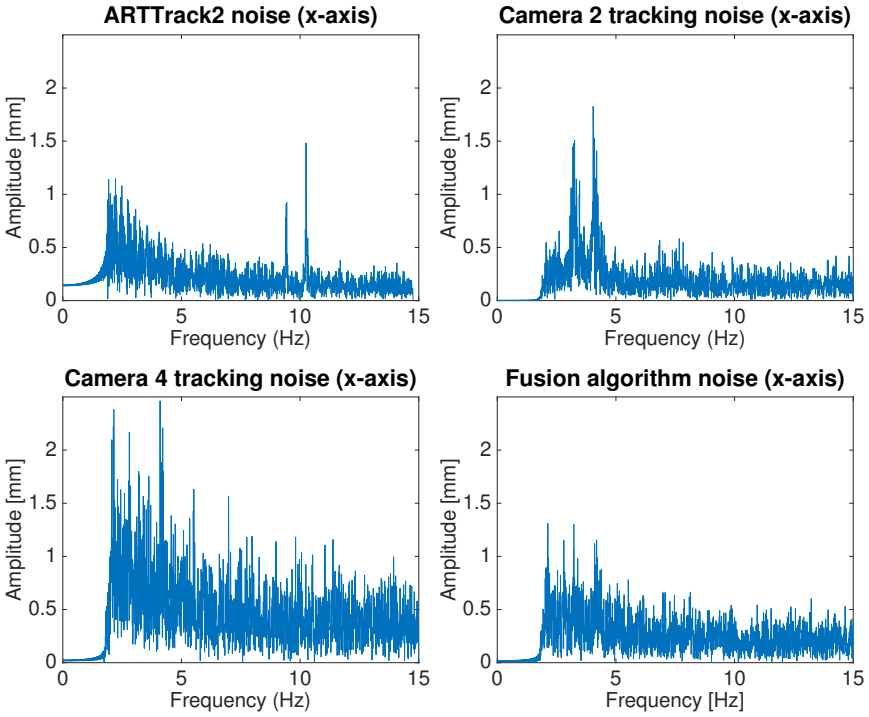


Figure 6.8.: Single-side amplitude spectra of the noise for the three skeleton tracking methods and the ground truth.

The standard deviation and the quartile range of the noise of all three methods and the ground truth is shown in table 6.8

Table 6.8.: Standard deviation and quartile range of the noise of the different tracking methods in mm.

Camera	Standard deviation	1st quartile	3rd quartile
ARTTrack2	7.37	-1.84	1.86
Camera 2	7.07	-5.29	5.47
Camera 4	15.34	-7.03	7.93
Fusion	8.38	-4.21	4.56

6.1.2.5. Performance evaluation

The results from the camera system performance evaluation as presented in chapter 6.1.1.5 are shown in the following. In the experiments both systems showed misclassifications in case of moving equipment, such as monitor stands and robots. These objects were occasionally classified as humans.

6.1.2.5.1. Kinect™ 360 system

Slow movement test: For test person one in all three iterations of each 3 minutes together, the system lost the detection of the human once for less than 2 seconds and once for less than 3 seconds. For test person two the system lost the detection of the human once for less than 2 seconds.

No movement test: The results from the test where the test persons stopped any movement are shown in table 6.9

Hand movement test: The results from the test where the test persons only moved their hands are shown in table 6.10

Detection time test: The results of the test for the detection times are shown in table 6.11

False-positive-classification test: For test person 1 in 12 of 13 iterations the system was able to distinguish between two humans even during contact. In the remaining sample the system detected both humans as a single human at a distance of 0.4 m. For test person 2 in 8 of 12 iterations the system was able to discriminate between two humans even on contact. In 1

sample both humans were detected as one human at a distance of 0.4 m. In two samples the distance was 0.1 m. In the remaining two samples the distance was 0.0 m (on contact).

Table 6.9.: Median and quartile range in sec for the time until detection is lost in the "no movement test" with the Kinect™ 360 system

Test person	Median Value	1st quartile	3rd quartile
1	14.0	11.0	16.8
2	15.0	6.0	17.3

Table 6.10.: Median and quartile range in sec for the time until detection is lost in the "hand movement test" with the Kinect™ 360 system

Test person	Median Value	1st quartile	3rd quartile
1	9.0	6.0	27.0
2	16.0	11.8	23.5

Table 6.11.: Median and quartile range in sec for the time until detection is present in the "detection time test" with the Kinect™ 360 system

Test person	Median Value	1st quartile	3rd quartile
1	1.0	0.0	1.0
2	1.0	1.0	1.0

6.1.2.5.2. Kinect™ One system

Detection time test: The results of the test for the detection times are shown in Fig. 6.9.

Performance test: For both test persons and each of the iterations the results are shown in table 6.12. The table shows the duration during which no human was detected though a human was present. Also the table shows the true-positive detection rate of the Fusion algorithm, which means that multiple point clouds from different views have been correctly classified as one and the same person. During the experiment, the system occasionally counted a number of

humans bigger than one though only one person was present within the FoV. This means that the Fusion algorithm classified multiple views of the same human as two humans, which is a false-negative classification. Using the true-positive and the false-negative values, the sensitivity of the fusion approach is computed and shown in the table. The system was able to track the human at any time it was able to detect him/her.

False-classification test: The results are shown in Fig. 6.10.

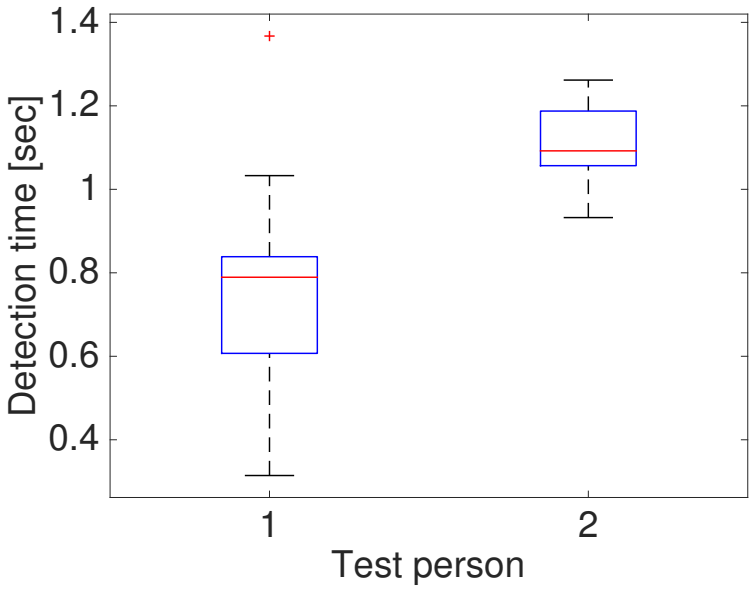


Figure 6.9.: Boxplot showing the detection time in seconds for the Kinect™ One cameras and both test persons.

Table 6.12.: Test results of the performance test for the Kinect™ One system. True-positive, false-negative and no detection are given in seconds. Sensitivity is given in percentage.

Person	Iteration	True-positive	False-negative	No detection	Sensitivity
1	1	178.83	1.17	0	0.99
1	2	165.80	14.2	0	0.92
1	3	176.17	1.43	2.4	0.99
2	1	175.4	2.45	1.33	0.98
2	2	176.47	2.8	0.73	0.98
2	3	159.43	20.57	0	0.89

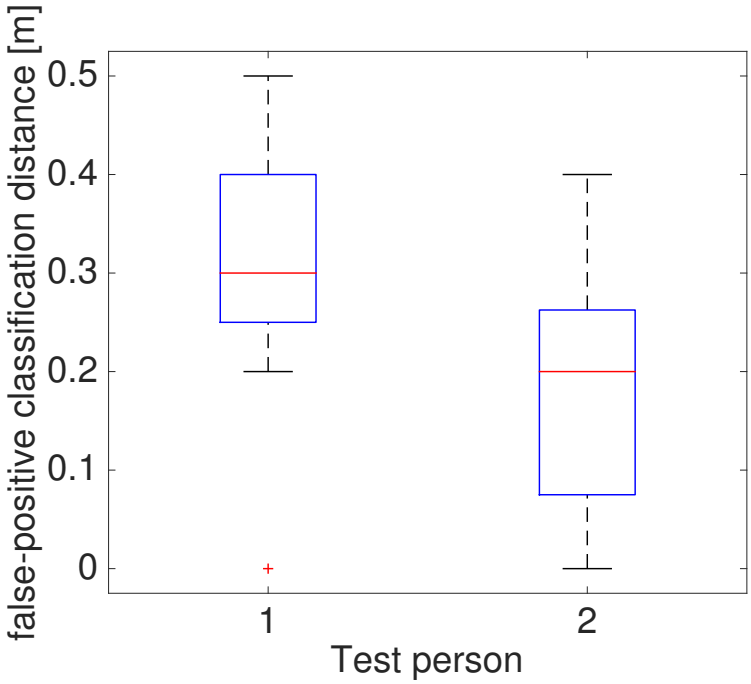


Figure 6.10.: Boxplot showing the distances between persons for which the Kinect™ One camera system classified two persons as a single person in the experiments.

6.2. Workflow-based controller

6.2.1. Experimental protocol

The workflow-based controller has been evaluated using the workflow shown in chapter 5.3. The experiments have been executed in a laboratory environment and an object transfer task has been executed, which replaced the phantom manipulation task (see Fig. 5.32). In this task, orange rings had to be picked up using the laparoscopic instruments that were inserted to an abdominal phantom via trocars. The rings had to be placed on metal sticks, which were also in the field of view of the camera as can be seen in Fig. 6.11.

9 test persons with a background in human machine interaction and robotics have been asked to perform the experiments and the workflow has been explained to them. They were told that the workflow includes the calibration and configuration of the system as well as the execution of the pick and place task and the extraction of the robotic system from the phantom. The visualization of the workflow has been explained to the test persons and the test persons were allowed to ask questions at any time. The current state of the process was visualized as a graph as shown in Fig. 5.24 on a television

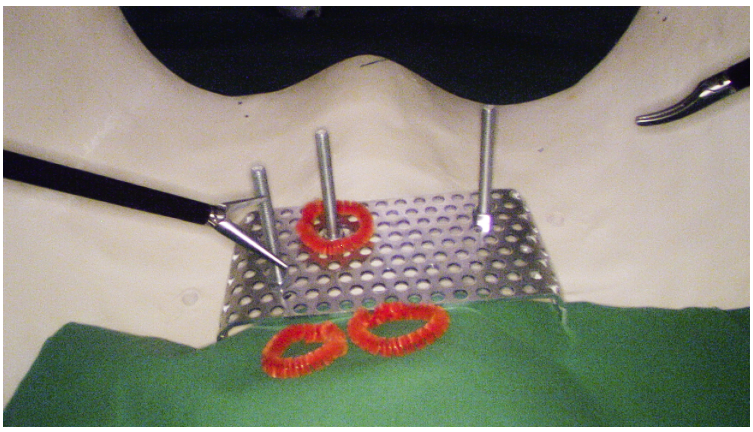


Figure 6.11.: The object transfer task as executed in the laboratory experiments.

screen that was visible to the test persons at any time. Assistance has been provided to the test persons in case they got stuck or got insecure in how to proceed with the execution of the workflow. During the execution of the workflow, the task activation time for each of the tasks has been logged by the workflow-based controller. The test persons were instructed to focus on how the system assists during the execution of the workflow and how they perceive the entire human-machine interaction, rather then on the execution of the task itself.

During the execution of the process, the Kinect™ One system was used and the Kinect™ 360 system was handled as an alternative component to the Kinect™ One system. Shortly before the execution of the task *telemanipulation_bimanual*, the Kinect™ One system was shut down without the knowledge of the test persons, so that the workflow-based controller considered the system as *unavailable* component and switched to the alternative Kinect™ 360 system.

After the experiments, each test person was asked to fill out questionnaires. As questionnaires the NASA-TLX (Task load index) [51], the UEQ (User Experience Questionnaire) [76] and a custom questionnnaire have been used.

The NASA-TLX is a subjective and multidimensional questionnaire to assess the perceived workload of a test person. It includes the six scales listed in the following:

- Mental Demand
- Physical Demand
- Temporal Demand
- Performance

- Effort
- Frustration

Each scale allows to choose one of 21 gradations on the scale. The pen and paper version, as can be seen in Fig. B.3 (Annex 2), has been used.

The UEQ is a subjective and multidimensional questionnaire to asses the quality of the user experience of software programs or other interactive systems. It contains 26 scales. Each scale is characterised by a pair of oppositional words in between which 7 gradations are available to the test person. The questionnaire can be seen in Fig. B.1 and Fig. B.2 (Annex 2). The analysis of the UEQ gives a score of the tested system on each of the following scales:

- Attractiveness
- Perspicuity
- Efficiency
- Dependability
- Stimulation
- Novelty

The score for the six final scales is computed using the protocol shown in Laugwitz et al. [76] from the results from the 26 scales the test person evaluated the system on. The score is then evaluated against the database of the UEQ, which contains results and statistics from prior experiments with systems evaluated using the UEQ.

Using the custom questionnaire, the test persons were asked if they noticed a failure or unexpected behavior during the execution of the process and at which point this has been noticed. The test persons were not told that one of the components was simulated to fail (Kinect^{TM} One system) and that its functionality has been taken over by an alternative component before they had completely filled in the questionnaires. Finally the test persons were asked for there general opinion about the system. This questionnaire can be seen in fig B.4.

6.2.2. Experimental results

The workflow-based controller has been evaluated using the protocol as shown in chapters 5.3 and 6.2.1.

As shown before, each of the test persons had a background in robotics and/or human machine interaction. Additionally, test persons 1, 2, 5 and 6 had experience in robotic telemanipulation.

Each of the test persons was able to successfully finish the workflow using the workflow-based controller. None of them faced major problems during the execution of the workflow.

During the execution of the workflow by the second test person, the capturing of the trocars as well as localization of the robot bases proved to be highly dependent on the position of the trocars and the robots' flanges with respect to the optical tracking system. Also occlusions seamed to heavily affect the accuracy of the optical tracking system. Due to the long transformation chains, small errors can heavily affect the accuracy of the overall system.

In order to bias and to distract the test persons as few as possible by a failure of the system, precomputed poses for the robots' base poses have been used for test persons 2-9. It has been communicated to the test persons that the poses for the robot bases have been precomputed and the result from the calibration steps within the workflow have been neglected.

During the execution of the workflow by test person 1, the system showed its ability to calibrate itself as part of the workflow, as in this first execution of the workflow no precomputed poses have been used.

As the trocar poses are not known prior to the execution of the workflow, they had to be captured during each of the test runs and it was not possible to use precomputed poses for the trocar. This resulted in failures for the test persons 4, 7, 8 and 9 that required to capture the poses of the trocars for a second time during the execution of the process. A possible solution to this problem is to use a pointing device that is mounted to the robot instead of the surgical tool. This allows to capture the trocar position in the coordinate frame of the end-effector of the robot, which reduces insecurities but requires to perform an additional tool change during the procedure.

6.2.2.1. Custom Questionnaire

The overall impressions of the test persons as shown from the custom questionnaire and the *further opinion* free text field mostly stated a high usability of the system. Two of the test persons did not use the free text field. Four of test persons used the free text field in order to show a positive impression of the system (e.g. *nice job*, *fancy*, *intuitive*). Three of the test persons suggested to improve the visualization of the workflow in terms of e.g. contrast, visibility or speaking task names. One of the test persons criticized that the vizualization does not show, when the system requires interaction by the operator. One

of the test persons expressed in a neutral manner that the focus was on the task execution and the implementation was mostly hidden. Other positive comments were related to the camera-based people detection, the telemanipulation system and the chosen workflow. Other negative comments criticized that the used workflow was mostly linear, or that the judgement about the depth from the 2D image of the endoscopic camera was cumbersome.

The free text field that was intended to be used in case of system failures was used by 6 of the 9 test persons. The other three test persons did not notice any failures of the system. 5 of the 6 test persons criticized failures of the system that were caused by the above shown problem of falsely captured trocar points. The question however was intended to assess whether the test persons noticed that the Kinect™ One system has automatically been replaced by the Kinect™ 360 system after the simulated failure. Only one test person raised an issue that was related to this topic as the Kinect™ 360 system failed to correctly capture a human, which resulted in a delayed hands-on mode detection.

6.2.2.2. Task activation times

The task activation times for a dual Intel Xeon E5-2637 V3 machine with 64Gb RAM are depicted in Fig. 6.12. The number of samples taken for the task in the order of the tasks as can be seen in the figure were: 8, 8, 8, 8, 8, 8, 8, 8, 29, 30, 31, 21, 16, 19, 9, 24, 9. It can be seen that the task activation time differs for the different tasks, which is backed by a p-value of $p<0.005$ that has been computed using a Kruskal-Wallis method. This is due to the fact that not the processing time but the entire system response time as well as the task execution times have been taken into account, which is explained in chapter 7.1.

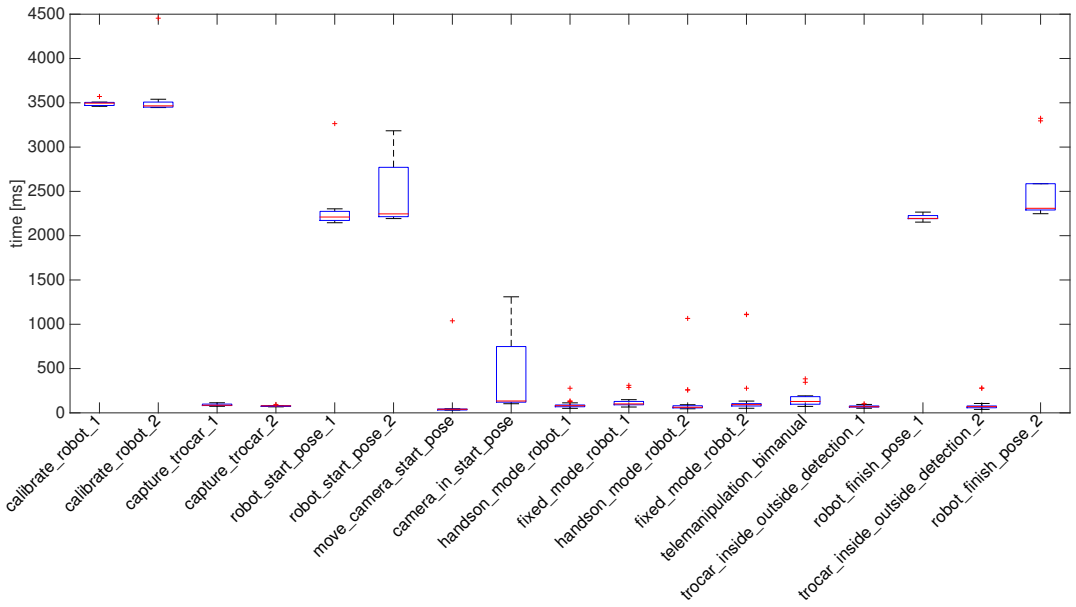


Figure 6.12.: Task activation times of the workflow-based controller for each of the tasks used in the experiments.

6.2.2.3. NASA-TLX Questionnaire

The results from the NASA-TLX questionnaire can be seen in Fig. 6.13 and table 6.13.

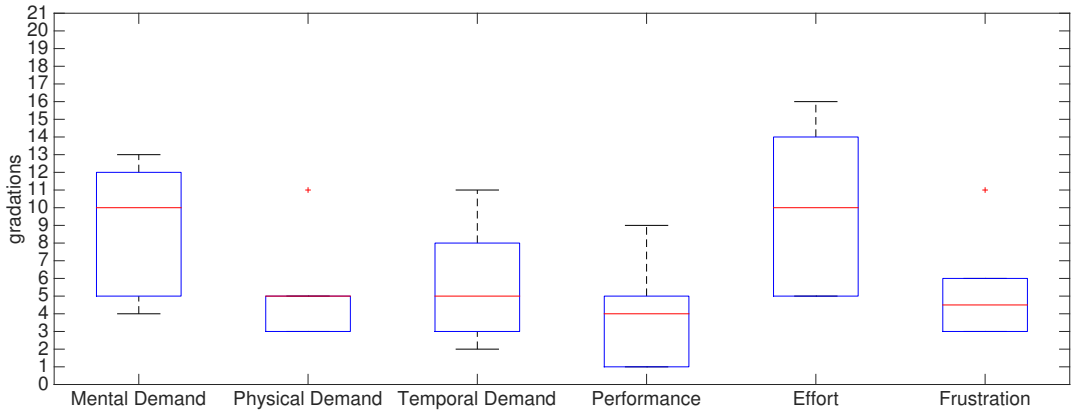


Figure 6.13.: The results from the evaluation using the NASA-TLX.

Table 6.13.: Median and quartile range of the NASA-TLX-based evaluation of the workflow-based controller in gradations

Scale	Median Value	1st quartile	3rd quartile
Mental Demand	10.0	5.0	12.0
Physical Demand	5.0	3.0	5.0
Temporal Demand	5.0	3.0	8.0
Performance	4.0	1.0	5.0
Effort	10.0	5.0	14.0
Frustration	4.5	3.0	6.0

6.2.2.4. UEQ

The UEQ transforms each of the scales with 7 gradations to a value range of -3 to +3, where -3 is worst and +3 is best. The UEQ then transforms the 26 scales to 6 measures that give an estimate about the user experience. The results for each item of the UEQ together with the word pairs (left,right) and the scales to which they contribute are shown in table 6.14.

Fig. 6.14 shows the results from the evaluation of the workflow-based controller on the resulting scales of the UEQ. It can be seen that each of the mean values are above 1 and therefore in the range of the results that are better than neutral. The same applies for the lower limits of the confidence intervals, with the exception of 0.911, which is the lower limit of the confidence interval for the scale *Dependability*. Using the benchmark included in the evaluation system of the UEQ, the results have been benchmarked against systems that have been evaluated with the UEQ prior to the evaluation performed during this work. The benchmark contains results from 163 studies and 4818 test persons. The results of the benchmark can be seen in Fig. 6.15 and table 6.15. It can be seen that the results from the evaluation for the scales *Attractiveness*, *Efficiency*, *Stimulation* and *Novelty* are in the range of the best 10% of the results. The results on the remaining scales *Perspicuity* and *Dependability* are better than 75% and worse then 10% of the results of the benchmark. It can also be seen that the results on each of the scales tend to be better than neutral.

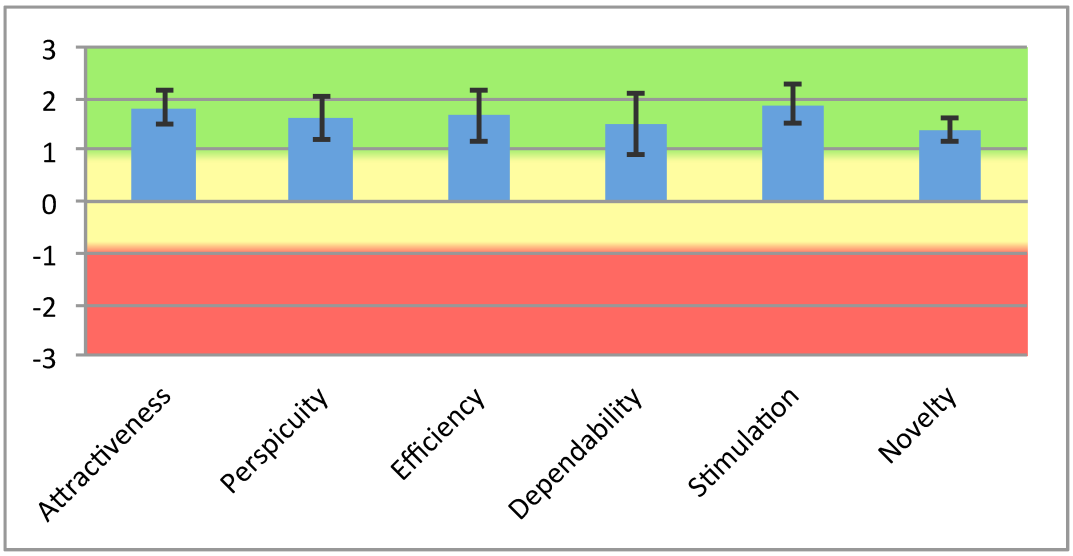


Figure 6.14.: The means (blue bars) of the 6 resulting scales of the UEQ-based evaluation with their confidence intervals (black). The background color signalizes how the results compare to a neutral evaluation. Green is better, yellow is neutral and red is worse.

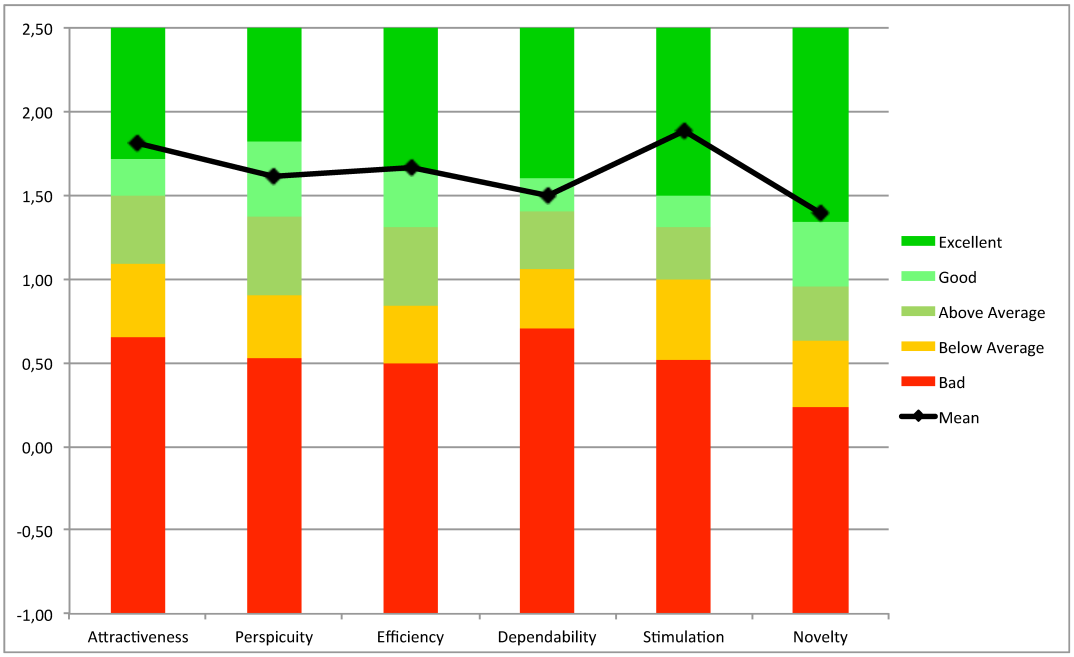


Figure 6.15.: The results of the benchmarking for the UEQ-based evaluation of the workflow-based controller. Black: Results of the evaluation. The color ranges of the bars are computed from the results of other systems that have been evaluated with the UEQ and are taken from the UEQ database.

Table 6.14.: The raw results from the UEQ-based evaluation of the workflow-based controller. The table shows the mean value of the transformed results for each scale, the standard deviation, the left and right values of the word pairs and the scale to which the item contributes. The arrow in the column mean gives an estimate on how the results differ from a neutral evaluation. Arrow up means better, arrow down means worse and arrow to the right means equal.

Item	Mean	Std. Dev.	Left	Right	Scale
1	↑ 2.0	0.5	annoying	enjoyable	Attractiveness
2	↑ 1.8	1.1	not understandable	understandable	Perspicuity
3	↑ 1.1	0.9	creative	dull	Novelty
4	↑ 1.6	1.1	easy to learn	difficult to learn	Perspicuity
5	↑ 1.8	0.8	valuable	inferior	Stimulation
6	↑ 1.7	0.9	boring	exciting	Stimulation
7	↑ 2.3	0.7	not interesting	interesting	Stimulation
8	↑ 1.4	1.0	unpredictable	predictable	Dependability
9	↑ 1.0	0.7	fast	slow	Efficiency
10	↑ 1.6	0.7	inventive	conventional	Novelty
11	↑ 2.2	0.4	obstructive	supportive	Dependability
12	↑ 2.4	0.7	good	bad	Attractiveness
13	↑ 1.7	0.7	complicated	easy	Perspicuity
14	↑ 1.6	0.5	unlikable	pleasing	Attractiveness
15	↑ 1.0	1.2	usual	leading edge	Novelty
16	↑ 1.8	0.7	unpleasant	pleasant	Attractiveness
17	→ 0.3	1.9	secure	not secure	Dependability
18	↑ 1.8	1.1	motivating	demotivating	Stimulation
19	↑ 2.0	1.0	meets expectations	does not meet expectations	Dependability
20	↑ 1.6	1.0	inefficient	efficient	Efficiency
21	↑ 1.4	1.6	clear	confusing	Perspicuity
22	↑ 2.0	1.1	impractical	practical	Efficiency
23	↑ 2.1	0.9	organized	cluttered	Efficiency
24	↑ 1.4	1.1	attractive	unattractive	Attractiveness
25	↑ 1.7	0.9	friendly	unfriendly	Attractiveness
26	↑ 1.9	0.6	conservative	innovative	Novelty

Table 6.15.: Comparison of the results from the UEQ-based evaluation to the benchmark with interpretation. The arrow shows how the mean value on the scale compares to a neutral evaluation. Up means better. Down means worse. Right means equal.

Scale	Mean	Comparison to benchmark	Interpretation
Attractiveness	↑ 1.815	Excellent	In the range of the 10% best results
Perspicuity	↑ 1.611	Good	10% of results better, 75% of results worse
Efficiency	↑ 1.667	Excellent	In the range of the 10% best results
Dependability	↑ 1.5	Good	10% of results better, 75% of results worse
Stimulation	↑ 1.889	Excellent	In the range of the 10% best results
Novelty	↑ 1.389	Excellent	In the range of the 10% best results

7. Discussion, Outlook and Conclusions

The following chapters discuss the findings and results of this work with relation to the aims as shown in the introduction chapter 1. Conclusions are given and future research directions are lined out.

7.1. Discussion

The results shown in chapter 6 show that both a markerless supervision system for the operating room and a workflow-based controller for surgical robots have been successfully implemented and evaluated. The results and findings from both perception systems and the workflow-based controller are shown in the following chapters.

7.1.1. Perception system

From the performance evaluation of both the Kinect™ 360 and the Kinect™ One system in chapter 6.1.2.5, it can be seen that the Kinect™ One system offers an improved performance over the Kinect™ 360 system. This is due to the improved depth acquisition performance of the Kinect™ One as the successor of the already excellent Kinect™ 360 camera. An additional performance increase is due to the use of the Kinect™ SDK for the Kinect™ One system instead of the OpenNI approach used for the Kinect™ 360 system. The Kinect™ SDK not only shows a higher detection rate for humans in the FoV of the camera but also allows to detect and track people using only one depth image that has been taken with the human in the FoV of the camera. In contrast, the OpenNI-based approach seemed to be dependent on movement and tracking was much more unreliable and unstable.

While the Kinect™ 360 system showed unstable full-body tracking, which did not allow for an evaluation, the tracking performance of the Kinect™ One system is of high quality and shows to be suitable for real life application. Additionally, the people detection capabilities of the Kinect™ One system

surpass the capabilities of the Kinect™ 360 system. From table 6.12 in chapter 6.1.2.5 it can be seen that the Kinect™ One system was able to detect the human in the FoV of the cameras in at least 98.7% of the samples in all test cases. These results can be regarded to be of very high quality, which is highlighted by the challenging test case that has been performed, where the test person completely stopped any movements. In comparison to the Kinect™ 360 system, which showed comparable results as long as people were moving, the Kinect™ One system did not loose track when people stopped moving or performed only hand movements. As can be seen in the detection time test both systems delivered comparable results for both of the test persons, which can be regarded to be low enough for real applications, especially when the stability of the detection and tracking of the Kinect™ One system is taken into account. The test also outlined the workspace of the Kinect™ One system, which exceeded the planned space workspace that only covered a small area around the operating table. Due to the stability of the Kinect™ One system, the results from the detection time test tend to be less important than they are for the Kinect™ 360 system as it is more likely that the latter permanently loses track of a human in the surrounding of the operating table.

In terms of time delay, the Kinect™ One system performed much better than the Kinect™ 360 system, whose time delay of 966 ms has been taken from literature [95]. The time delay in this work has been evaluated for both the skeleton tracking information and for the point cloud-based feature computation, whereas the latter has to be compared to the delay of the Kinect™ 360 system. Compared to the Kinect™ 360 system, the delay of the Kinect™ One system was found to be 287 ms (adjusted by the delay of the ground truth) and therefore only shows 29.7% of the delay of the Kinect™ 360 system. It is presumed that the cause for this massive decrease in the delay is mainly the lower network load for the Kinect™ One system compared to the Kinect™ 360 system, which results from the use of 10 Gb Ethernet. Additionally, it is presumed that the decentralized point cloud computation and skeleton tracking may help to decrease the delay and that the Kinect™ One system offers a lower acquisition delay compared to the Kinect™ 360 system. The delay of the skeleton tracking has been found to be even smaller with a delay of 88 ms (adjusted by the delay of the ground truth). The low delay shows that the system may be used for collision detection or prevention tasks in case of slow moving objects. For fast moving objects or high speed object tracking the system delay was still to high. With respect to the aims of this work the delay was found to be sufficient to accurately track people and to infer about their motions and actions without a disturbance of the user experience.

In terms of sensitivity, the fusion approach for the Kinect™ One system shows very high values of 89-99%, which show that most of the time the

system is able to find corresponding point cloud representations of one and the same human. However, if humans are standing close to each other the system may not be able to discriminate between them, as can be seen in chapter 6.1.2.5. This is an indicator for an improvable specificity and the only discipline where the Kinect™ 360 system is superior to the Kinect™ One system. The effects of false-negative and false-positive samples from the fusion algorithm can lead to wrong or inaccurate full body tracking or features. The proximity of actors is a common situation in the operating room and the reduction of false-positive classifications is highly desirable and can either be achieved by decreasing the threshold for the user fusion algorithm, which will result in a lower sensitivity, or by replacing the centroid-based correspondence detection by measures that produce lower position errors and therefore to decrease the threshold without losing sensitivity. Additionally, a reduction of the registration errors can help to reduce the required value for the threshold. In order to achieve this aim, future research could involve the improvement of the registration by using refinement methods, such as the ICP to decrease the registration error of the point clouds. In order to tackle the problem of misplaced centroids and partial views, it may be possible to perform a model-based reconstruction of the human from partial views, which can help to extract a measure that is always in the center of the human, which gives the possibility to reduce the thresholds. It may also be possible to segment body-parts of the humans from the point clouds (e.g. head) and to assess if they overlap when point-clouds from multiple cameras are used and a fusion is intended to be performed.

The results from the registration of the system components with respect to each other show a sufficient registration quality for the application and for all of the used algorithms. The pairwise registration algorithm shows a good registration quality of the point clouds from Kinect™ 360 cameras to each other. In this scope, the median of the worst registered camera was 26.89 mm. From the statistical analysis, it can be seen that for one of the Kinect™ 360 cameras there was a statistically significant lower registration error compared to the other two cameras that have been registered. This camera was located closer to the reference camera compared to the other two cameras and it is presumed that the algorithm performs better when the cameras are in proximity.

Compared to this, the results from the algorithm to register a camera to an external reference were inferior to the results gained through the pairwise registration algorithm. The distribution of the errors in the different pairs of cameras was much more inhomogeneous compared to the results from the pairwise registration algorithms, as can be seen from the p-values and the boxplots 6.2 and 6.3. Additionally, the overall error level was higher.

These results are in contrast to the results from the camera registration eval-

uation in experiments 3 and 4, which tend to indicate that the algorithm to register cameras to an external reference performs better than the pairwise registration algorithm. This can be explained by the relatively small workspace in which the cameras can be calibrated using the calibration algorithm to calibrate to an external reference, which is due to the small overlapping field of views of the cameras and the optical tracking system. This leads to a good overlap in the area where the system has been calibrated, but small rotational errors can lead to big positional errors at locations farther from the area in which the system has been calibrated. However, it is assumed that the depth processing error of the Kinect™ 360 cameras, which is up to 75 mm at a distance of 3.5 m according to Karan et al. [62] introduces an even bigger error compared to the error introduced due to a small calibration area. This means that for uncalibrated Kinect™ 360 cameras a good registration of the IR cameras does not necessary lead to a good registration of the point clouds.

If a lower error is required, a solution is to perform either a depth calibration as also shown in the work of Karan et al. or to perform a registration for the 2D IR images and an additional registration for the generated point clouds using an ICP algorithm or the methods from the pairwise registration algorithm.

Finally, in the case of the algorithm to register cameras to an external reference, the registration matrix between two cameras is computed from two registration processes. This may lead to an increased or a decreased error, which cannot be seen from the collected data.

The registration error of the IR images of the Kinect™ 360 cameras to the reference is smaller compared to the error of the Kinect™ One cameras. The latter use higher quality cameras, but also have a bigger field of view. This means that a smaller image area on the camera sensor is used for registration, which leads to bigger registration errors. All together the registration errors of the IR cameras of both systems to the external reference are smaller than the point cloud registration error as evaluated in experiments 1 and 2. For both camera systems the registration error distribution for the used cameras are inhomogeneous, if the algorithm to register a camera to an external camera is used. The cause for this is unknown.

An estimation about the depth registration error of the Kinect™ One system can be made, when a look is taken on the positional error of the skeleton tracking as shown in chapter 6.1.2.4. The median error for camera 4, which was inferior to the other evaluated camera, was 59.48 mm which is on a similar level compared to the depth registration error for the Kinect™ 360 system, when the algorithm to an external reference is used. From this, an assumption about a possible depth error of the Kinect™ One camera cannot be made as the skeleton tracking algorithm can possibly introduce an additional error

and as the reference markers of the optical tracking system cannot be placed inside the human joint that is tracked by the Kinect™. In the experiments, the markers have been placed on the back of the test persons hand, which introduces a systematic error.

The position error from the skeleton tracking shows the final tracking error using the proposed system. The data also shows that the skeleton fusion approach averages the position error of the skeleton tracking data that has been used for fusion. This means that the position error from the fusion approach lies between the position errors of both data sources/cameras, which in the experiments showed a median value of 49.13 mm as can be seen in chapter 6.1.2.4.

In comparison to the registration errors of the cameras, the robot localization error is relatively small and showed a median value of 3.50 mm.

The results from the noise evaluation show that the noise level of the skeleton tracking is higher than the noise level of the ground truth data. The noise level of both used Kinect™ One cameras was different and one of the cameras showed a higher noise level, which can not explained by using the available data. The noise level from the fusion approach lies between the noise levels of the used cameras and is close to the noise level of the camera with the lower noise level. It can also be seen that amplitude peaks in the frequency domain of the noise are smoothed as can be seen from the amplitude spectra, which improves the tracking quality compared to a single camera approach.

This means that in terms of noise and in terms of position error the fusion approach delivers better results than the raw skeleton tracking data from the Kinect™ One cameras. Also the reliability is higher as tracking can continue if one of the cameras looses track of the human. In any case the fused tracking information should be preferred over the raw tracking data.

A comparison with the state-of-the-art shows that other researchers put focus on the actions performed at the patient side rather then actions performed in the surrounding of the operating table. A similar system to the one shown in this work is described in the work of of Ladikos et al. [72], which is intended for collision avoidance and shows a lower delay of 25 msec compared to the delay of 106 ms as evaluated before. The system then has been used for successful workflow monitoring as shown in the work of Padoy et al. [100]. Their system is not able to perform body tracking and limits the data analysis to the voxel-based representation. In the work shown in [97], the authors evaluated the positional tracking error of the Kinect™ 360 camera using motion capturing systems as ground truth. They observed mean distances between 63 mm and 75 mm for the hand wrists. The mean error for the hand position as achieved in this work is 57.53 mm and is therefore lower compared to the work of [97]. Other works focused more on registration and calibration

of RGB-D cameras as shown in the work of Karan et al. [62], which was not the main scope of this work.

To the best of the author's knowledge there are no other systems that

- allow to monitor the surgical operating room with people tracking capabilities
- allow to fuse and utilize both skeleton tracking and point clouds
- evaluate the use of Kinect™ 360 and One against each other in a multi camera setup
- utilize a camera system to focus on the actions in the surrounding of the operating table to control a surgical operating room.

With respect to the goal of this work, the perception system fulfills all requirements for tracking and detection of surgical personnel in the operating room. The detection and tracking quality is high and allows to serve as input data for situation analysis systems, such as the workflow-based controller shown in this work. Though the registration error is low enough for the application and the algorithms have proved to cope with the registration error, improvements can be made if a higher registration accuracy is achieved. This may be useful if the system is considered to be applied to other possible areas of application. Such areas may be the supervision of production plants, The volumetric measurement of humans or objects, or the human gait analysis.

Other improvements can be made in terms of specificity of the fusion approach or in terms of the used camera. For an application in the operating room, fanless cameras or a sealed enclosure are necessary. It is also of highest interest, which findings can be made about the set of detectable actions in the operating room by the use of such a system. For the workflow tracking and workflow-based control of operating rooms, a large set of detectable situations is required to increase the system performance.

7.1.2. Workflow-based Controller

As can be seen in chapter 6.2, each of the test persons were able to execute the test case shown in chapter 5.3. The entire system was driven by the knowledge and none of the connections between the workflow and the physical system had to be programmatically designed. This demonstrates the feasibility of knowledge-based execution of workflow-guided interventions in heterogeneous distributed applications.

Although the system has been defined with surgical robotics in mind, the actual implementation is generic and therefore each of the findings can not

only be applied to the execution of workflow-guided interventions using robotics but also to the context-aware control of surgical intervention theatres in general.

The wide range of implemented tasks in the field of calibration, registration as well as medical tasks show the flexibility of the approach. The designing method for the workflow is easily understandable and can be adapted to be used by domain experts such as surgeons or scrub nurses in order to adapt the workflow to the needs of the personnel, patient or the procedure. One of the most important finding is that the implemented concept allows the personnel to calibrate and use a highly complex system during the execution of application-driven workflows without deep knowledge about the actual implementation of the system. It is even more important that the personnel is provided with the best available components for each task to be executed.

During the experiments it has been successfully demonstrated that the concept allows to cope with changing conditions in the operating room. The system was not only always able to keep track of the workflow to be executed, but also facilitates the compensation of failing or missing components. This has been demonstrated using a simulated failure of the Kinect™ One system that has been replaced with the Kinect™ 360 system.

A control-strategy for the execution of workflows on heterogeneous systems has been successfully developed and integrated into the workflow-based controller, which selects and configures components based on the needs of the tasks to be executed and then, during the execution of the task, monitors the data from the system and compares it with reference values to keep track of the workflow.

The perception system developed in this work and the telemanipulator developed alongside with this work have been completely interfaced with the workflow-based controller. Due to the variety of the tasks, as expected, the perception system developed in the scope of this work solely is not sufficient for the monitoring of the entire workflow. However, it has been shown that it helps to improve the interaction of the personnel with the system by taking the environment of the operating field into account. This allows to give appropriate assistance in case environmental changes affect the workflow. It highlights that the introduction of new sensor concepts into the operating room help to make the surgical environments more flexible and help to increase the autonomy of assistance systems.

The analysis of the activation times of the workflow-based controller show that some tasks can be activated in well below under 1 sec while others require >4 sec in order to be activated. This is due to various reasons. The *dynamic_reconfigure* mechanism used for the configuration of the system is a synchronous call. This means that the workflow-based controller will wait

for the call to return. Based on the implementation of the configuration on the component side, the call can return immediately or after the actual execution of the task. This can be used to make the workflow-based controller wait for e.g. autonomous robot movements but can also be used to initiate longer lasting that run in parallel with other tasks. Activation times are further affected by network load, response times of the components and finally the number of components to be used and configured. It can be seen that in the test scenario, tasks that initiate autonomous movements of robots typically require longer activation times than other tasks, which require activation times around 100 msec. The activation times do not introduce a noticeable delay into the execution of the workflow by the personnel. This is confirmed by the scale *fast-slow* of the UEQ where the results show a mean value of 1.0, which is well above the neutral result of 0.0. This means that in the current stage, the implementation already performs fast enough to avoid the introduction of noticeable delays into the workflow execution.

The evaluation of the NASA-TLX and the UEQ questionnaire showed excellent results in terms of user experience for the relatively small test group. Due to their experience, a bias of the test persons can not be fully excluded. The results also indicate that the test persons not only evaluated the interaction concept but also the task execution itself, which allows to evaluate the workflow-based controller concept using the implemented workflow, but does not necessarily allow to generalize the results.

As stated in the introduction, it is desired to decrease the workload of the personnel to allow them to focus on the actual intervention to be performed. The evaluation of the NASA-TLX showed that on scales *Mental Demand*, *Physical Demand* and *Temporal Demand* in median the test persons experienced a demand in the lower half of the range of the TLX. The test persons also felt to perform well and the frustration level was very low. Some of the test persons experienced a high effort during the execution of the workflow, while in median the effort has been felt to in the lower half of the range. This means that in median on every scale the NASA-TLX indicates results that are in the lower half of the ranges. These results show the high potential of the approach and support the statement that workflow-driven context-aware systems may help to decrease the workload.

The results from the UEQ-based evaluation of the system show that the test persons perceived the system to perform excellent. The benchmarking results indicate that on each of the scales the results, gained during the study performed with the system, are better than 75% of the results in the database of the UEQ. On 4 of the 6 scales the results are even in the 10% of the best results compared to the database. The results from the 26 word pairs in the questionnaire indicate that on 25 of the 26 scales the user felt the system

performs better than neutral. An exception for this was the word pair secure-insecure, where the results were only slightly above neutral.

Outstanding results have been achieved for the word pairs *not interesting-interesting*, *obstructive-supportive*, *good-bad* and *organized-cluttered*, where the value of 2.2 for the word pair *obstructive-supportive* is of highest importance as the test persons perceived to be very well supported, which was one of the primary goals of the work.

As can be seen from the results, only one of the test persons was distracted from the task due to a simulated failure of the camera system. This person noticed the decreased detection system performance, although the system remained operational. None of the test persons was able to perceive that there was a major (simulated) failure of a component during the execution of the workflow. This emphasizes the systems supportive capabilities as mild and recoverable errors can be entirely tackled by the system without or little distraction of the personnel, which allows them to stay focused on the task.

The aims as shown in the introduction have been entirely achieved. The results underline the high quality of the concept and its potential to improve existent and future assistance technologies. Especially the results from the UEQ emphasize the excellence of the methods shown in this work. It is therefore of highest importance for future works to implement the concept for a real operating theatre and to evaluate the concept in medical studies with a purely medical target group and a sample size that is large enough to show that the excellent results are statistically significant for the target scenarios.

Possible applications for the concept are the improvement of assistance systems that are already in the operating room, image-guided surgery, or a complete integration of operating theatres. Apart from this it may be beneficial for industrial applications as the approach allows to rapidly implement a workflow for unknown target systems or to adapt the workflow to known target systems.

The approach shown in this work has shown excellent results in lab settings and may introduce new research directions in the relatively young and unexplored field of workflow-based context-aware systems in surgery. While workflow-based control has already been successfully used in industry, the purely knowledge-based abstraction between the workflow and the execution system is a fundamental modification of established paradigms, where usually a strict relation between workflow and workflow-execution is implemented.

In this scope, the most significant differences between the shown work and the state-of-the art can be identified. The works that are closest to the addressed topics can be found in [131], where a knowledge base has been used to assist the execution of workflows in case deviations from a preplanned workflow

are required using the example of oil spill recovery. Most other works like [100] or [73] show how surgical workflows can be detected and analysed. These works can be used complementary to this work but show very little similarities with the proposed approach.

To the best of the author's knowledge there is no other existing work in which workflows have been successfully used to compose and configure (medical) systems, whose structure is unknown prior to the run-time, by purely using knowledge.

7.2. Outlook

The methods shown in this work provide a basis for various future research directions, such as

- **Evaluation in the clinical scenario:** Future research should further pursue the concepts developed in this work in order to apply them to real medical applications and to evaluate their feasibility in surgical centers.

- **Knowledge integration in hospitals:** A big advantage of the shown approach is the modelling of the knowledge about the components used during a procedure. Future work could focus to integrate the developed concepts with already existent hospital information systems. This may allow for improved resource planing for operations and could also help to improve the workflow-based controller concept, as the patient information can contribute to the process execution. This means that the scheduling of operations could be improved or that special needs of patients could be considered by the workflow-based controller in order to further improve the procedure.

- **Workflow-management enhancements:** As shown in this work, up to now, technical and medical aspects are mixed in the workflow. This means that although only few knowledge about the system to execute the workflow on is expected from the person in charge of the modelling of the workflow, he/she has to think about the technical feasibility of the workflow. Future research could involve a secondary abstraction layer, which allows to translate a purely medical workflow to the existent mixed technical and medical workflow, which further decreases the workflow modelling effort. This could e.g. be implemented by using YAWL's composite task mechanisms.

- **Semi-autonomous knowledge engineering by machine learning:** In the current implementation, the entire knowledge in the knowledge-base of the workflow-based controller has been modelled by a human. This forms the basis for the application of machine learning technologies. It may significantly improve the system capabilities if parts of the knowledge in the knowledge base could be learned from observations. This would e.g. allow to apply a workflow even faster to a target system and to quickly integrate new components. Furthermore, it may allow to identify aspects in the workflow that are not striking at first sight but may help to improve the execution of the actual tasks within the workflow. The workflow-mining capabilities of YAWL could serve as a base for such technologies.
- **Perception system enhancements:** Despite the decent results, the detection and registration quality of the system could be improved and it may also be desirable to decrease the delay and to increase the frame-rate for high speed applications. Additionally, prediction methods for human movements in the FoV of the cameras could allow to compensate for short detection losses. Using the RGB image, it may also be possible to identify the humans in the FoV of the system. This could allow to detect role-dependent actions in the operating room.
- **Application of perception system to other areas:** The methods used for the perception system could be applied to perception systems in industrial areas to allow for a safe-human robot interaction in manufacturing. Additionally, the methods could be used for indoor and outdoor supervision tasks of critical areas or for the interaction with virtual reality applications.
- **Technological feedback to industry:** The concept as described in this work originated in Business Process Modeling. Industrial applications tend to become integrated and networked systems in which knowledge processing plays a major role. The knowledge-based translation of workflows to a target system as shown in this work could help to improve and to accelerate Business Process Modelling and workflow-driven applications in industry.

7.3. Conclusions

In this doctoral thesis, a complete system for the workflow-based execution of surgical procedures has been designed, a prototype of the approach has been implemented and the approach has been qualitatively and quantitatively evaluated. Thematically, the work can be seen in the scope of workflow

analysis and workflow detection. Despite various works in this field have been published, workflow modelling and tracking in the operating room is still in its beginnings. This work describes a complete solution that covers the entire process chain for workflow-based surgical procedure execution including the modelling, execution and tracking of the workflow. Special to this work are the knowledge-based transition of the workflow to the target system, the dynamic system composition and the sensory system that includes environmental information in order to enhance the feature space for the tracking of the workflows.

The approach facilitates the knowledge-based execution of a procedure, which has been exemplarily shown at the example of telemanipulated surgery. In this context, the workflow-based control and its environment perception system demonstrated to show a decent user experience, which is confirmed by excellent results from the performed study. In the UEQ benchmark, the results from the study state that the results on 4 of 6 scales are better than 90% of the results from the benchmark and on 2 of the 6 scales are better than 75% of the results from the benchmark.

The scientific contributions of this work are listed in the following:

- **Environmental supervision for the operating room:** The perception system developed in the scope of this work shows a large workspace, a high people tracking and detection quality as well as a registration error and a delay that are small enough to facilitate high quality people tracking and workflow detection. Compared to prior art, the system shows a high robustness and can provide a point cloud representation for each of the humans in the field of view, as well as a low-noise full-body tracking simultaneously, while being able to discriminate between multiple humans in the field-of-view.

- **Execution of surgical workflows:** The developed workflow-based controller allows for a direct execution of graphically defined workflows in a laboratory environment. This is achieved by a knowledge-based workflow management system that automatically selects suitable components to keep track of the execution state of active tasks. This is unique to the system and facilitates the system to rapidly integrate surgical workflows or to adapt them to the needs of the procedure, surgeon or patient.

- **Generic handling of surgical devices:** The knowledge-base of this work has been designed to facilitate generic handling of devices. This allows to adapt the approach to other target areas within medicine, like imaging, or non-surgical treatment and offers the possibility to transfer the system to applications beyond medicine.

- **Knowledge-based execution of workflows on dynamic and uncertain target systems:** A major point in the development of this concept was the knowledge-based translation of workflows to target systems. Novel to the approach is that any connection between the workflow and the execution of the workflow is created from knowledge during the run-time of the workflow. This allows to select suitable components for the execution of the task during the run-time of the workflow and facilitates to adapt to a dynamic or changing system. This can greatly help to improve the execution of the task

- **Fault tolerant execution of surgical workflows:** The knowledge-based selection during the run-time of a workflow not only allows to select components and to compose the execution system but also facilitates to check the availability of components and to compensate for missing or failing components.

- **Natural interaction with highly integrated systems:** Despite not being a main topic for the development of the concept the telemanipulation study showed good to excellent results in terms of user interaction with the system. This raises the topic of natural user interaction to which the approach can contribute. As the system is able to follow a workflow that is executed, it can be seen as a user interface, which acts context-aware and which has been perceived positively by the test persons.

- **Modular surgical robotics:** The shown approach benefits from highly modular subsystems that are to be controlled by the workflow-based controller. The developed workflow-based controller not only shows a way how to control, connect and integrate such systems but also contributes to the research on modular robotic systems in medicine to allow for the development of sub-systems that can include task-dependent components to increase the task-specific performance of the system. The research within this scope massively influenced the modularization strategy and the development of the OP:Sense framework, which was introduced in [94].

To the best of the author's knowledge, the proposed approach shows the first fully integrated workflow-based and context-aware controller for surgical systems that is entirely driven by knowledge. Compared to other works, the proposed approach is highly flexible, adaptable and opens research directions on workflow-based and context-aware control as well as environment monitoring for the improvement of medical systems. The character of the work tends to be basic research and provides a new perspective on workflow-guided systems in medicine. This can also be seen as the biggest drawback of the work, as up to now the concepts have not been brought to a real operating

room. Therefore the impact of the concept on clinical practice should be evaluated in future research.

In conclusion the work shows a workflow-based controller for surgical operating theatres that allows to integrate, interface and to connect various subsystems. It allows to increase the system performance, usability, flexibility and safety of highly integrated or robot-equipped operating theatres.

Appendix

A. Annex 1

Registration errors per checkerboard position for the different experiments as shown in chapter 6.1.

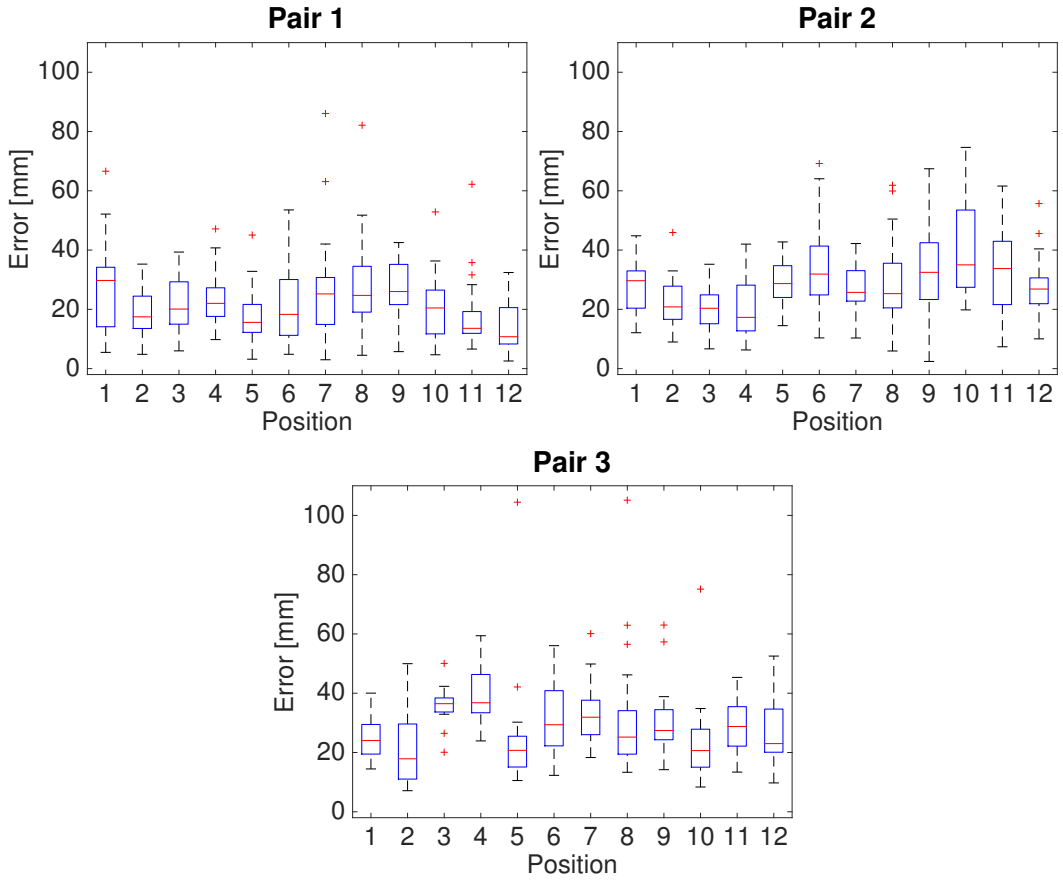


Figure A.1.: Boxplot showing the point cloud registration error in mm for each each of the checkerboard positions used to evaluate each pair of Kinect™ 360 cameras using the pairwise registration algorithm

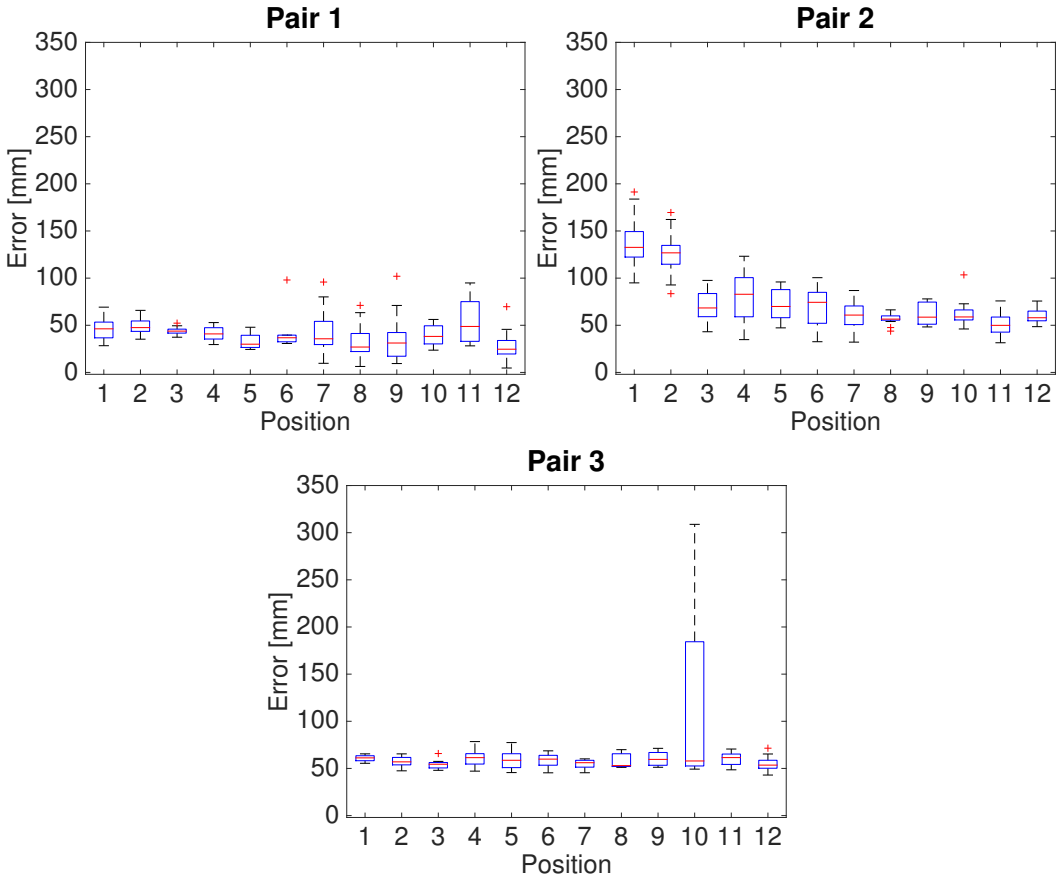


Figure A.2.: Boxplot showing the point cloud registration error in mm for each each of the checkerboard positions used to evaluate each pair of Kinect™ 360 cameras using the algorithm to register a camera to an external reference

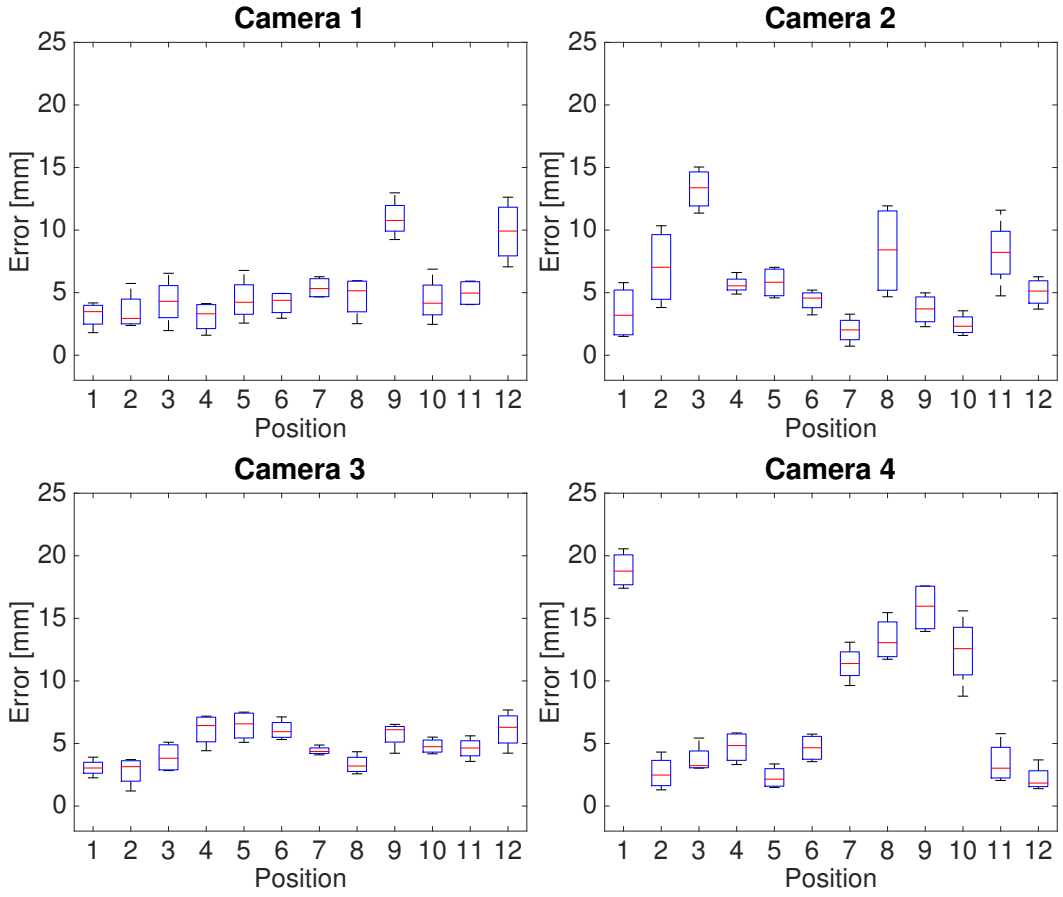


Figure A.3.: Boxplot showing the registration error in mm for each of the checkerboard positions for each Kinect™ 360 camera. The cameras are registered using the algorithm to register a camera to an external reference

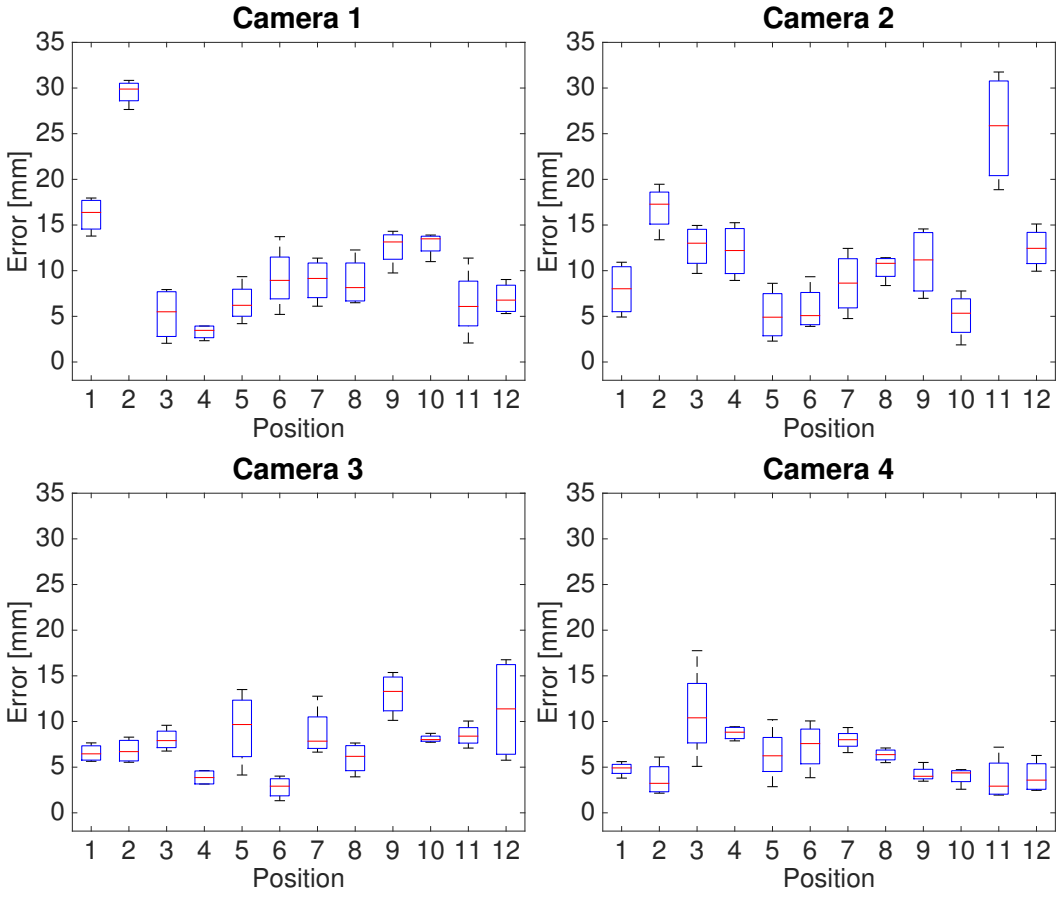
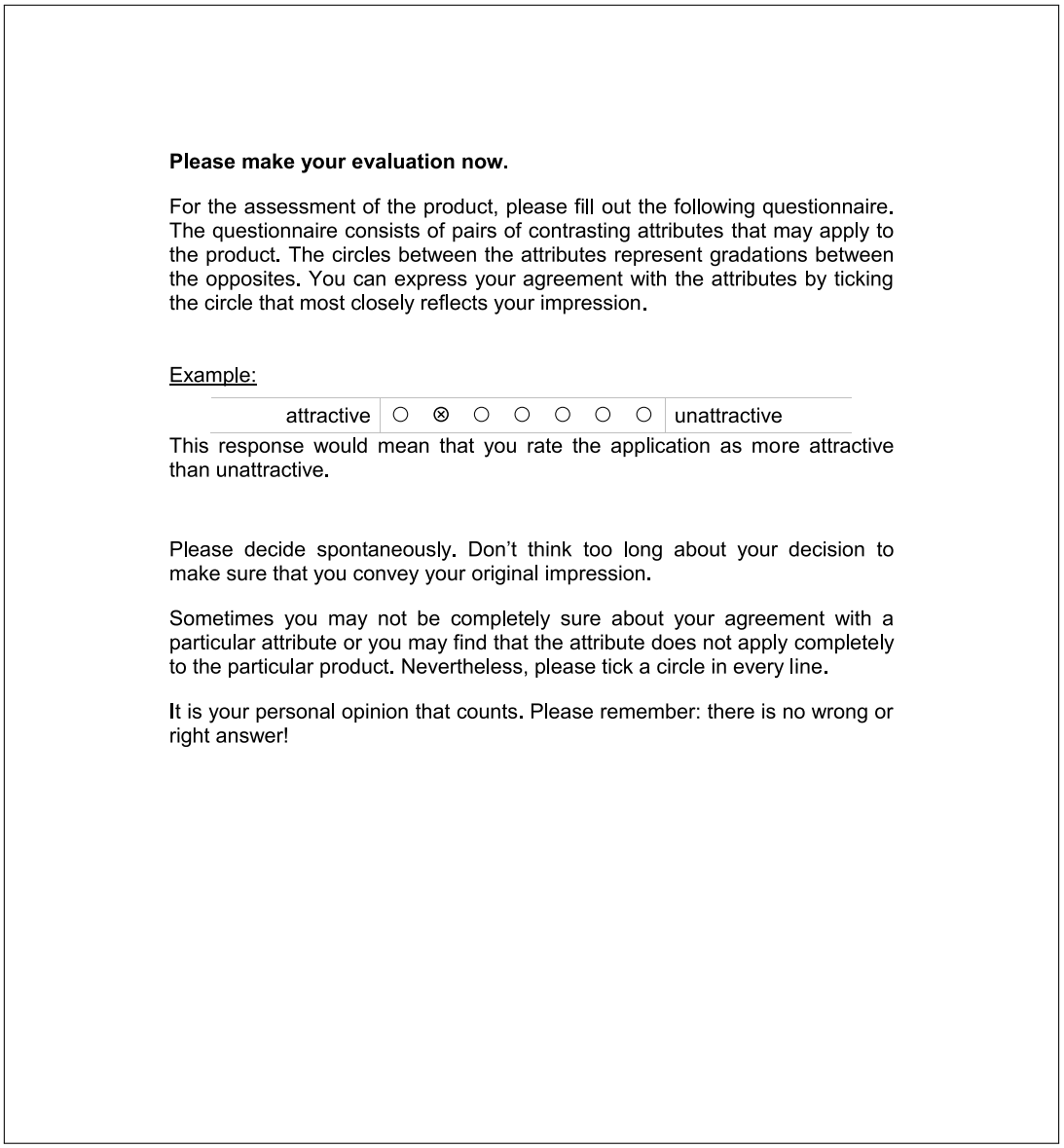


Figure A.4.: Boxplot showing the registration error in mm for each of the checkerboard positions for each Kinect™ One camera. The cameras are registered using the algorithm to register a camera to an external reference

B. Annex 2

The questionnaires used in the experiments as shown in chapter 6.2.



Please make your evaluation now.

For the assessment of the product, please fill out the following questionnaire. The questionnaire consists of pairs of contrasting attributes that may apply to the product. The circles between the attributes represent gradations between the opposites. You can express your agreement with the attributes by ticking the circle that most closely reflects your impression.

Example:

attractive	○	⊗	○	○	○	○	○ unattractive

This response would mean that you rate the application as more attractive than unattractive.

Please decide spontaneously. Don't think too long about your decision to make sure that you convey your original impression.

Sometimes you may not be completely sure about your agreement with a particular attribute or you may find that the attribute does not apply completely to the particular product. Nevertheless, please tick a circle in every line.

It is your personal opinion that counts. Please remember: there is no wrong or right answer!

Figure B.1.: Page 1 of the Pen/Pencil version of the UEQ. Source: [7], ©[2015] UEQ-Online

Please assess the product now by ticking one circle per line.

	1	2	3	4	5	6	7		
annoying	○	○	○	○	○	○	○	enjoyable	1
not understandable	○	○	○	○	○	○	○	understandable	2
creative	○	○	○	○	○	○	○	dull	3
easy to learn	○	○	○	○	○	○	○	difficult to learn	4
valuable	○	○	○	○	○	○	○	inferior	5
boring	○	○	○	○	○	○	○	exciting	6
not interesting	○	○	○	○	○	○	○	interesting	7
unpredictable	○	○	○	○	○	○	○	predictable	8
fast	○	○	○	○	○	○	○	slow	9
inventive	○	○	○	○	○	○	○	conventional	10
obstructive	○	○	○	○	○	○	○	supportive	11
good	○	○	○	○	○	○	○	bad	12
complicated	○	○	○	○	○	○	○	easy	13
unlikable	○	○	○	○	○	○	○	pleasing	14
usual	○	○	○	○	○	○	○	leading edge	15
unpleasant	○	○	○	○	○	○	○	pleasant	16
secure	○	○	○	○	○	○	○	not secure	17
motivating	○	○	○	○	○	○	○	demotivating	18
meets expectations	○	○	○	○	○	○	○	does not meet expectations	19
inefficient	○	○	○	○	○	○	○	efficient	20
clear	○	○	○	○	○	○	○	confusing	21
impractical	○	○	○	○	○	○	○	practical	22
organized	○	○	○	○	○	○	○	cluttered	23
attractive	○	○	○	○	○	○	○	unattractive	24
friendly	○	○	○	○	○	○	○	unfriendly	25
conservative	○	○	○	○	○	○	○	innovative	26

Figure B.2.: Page 2 of the Pen/Pencil version of the UEQ. Source: [7], ©[2015] UEQ-Online

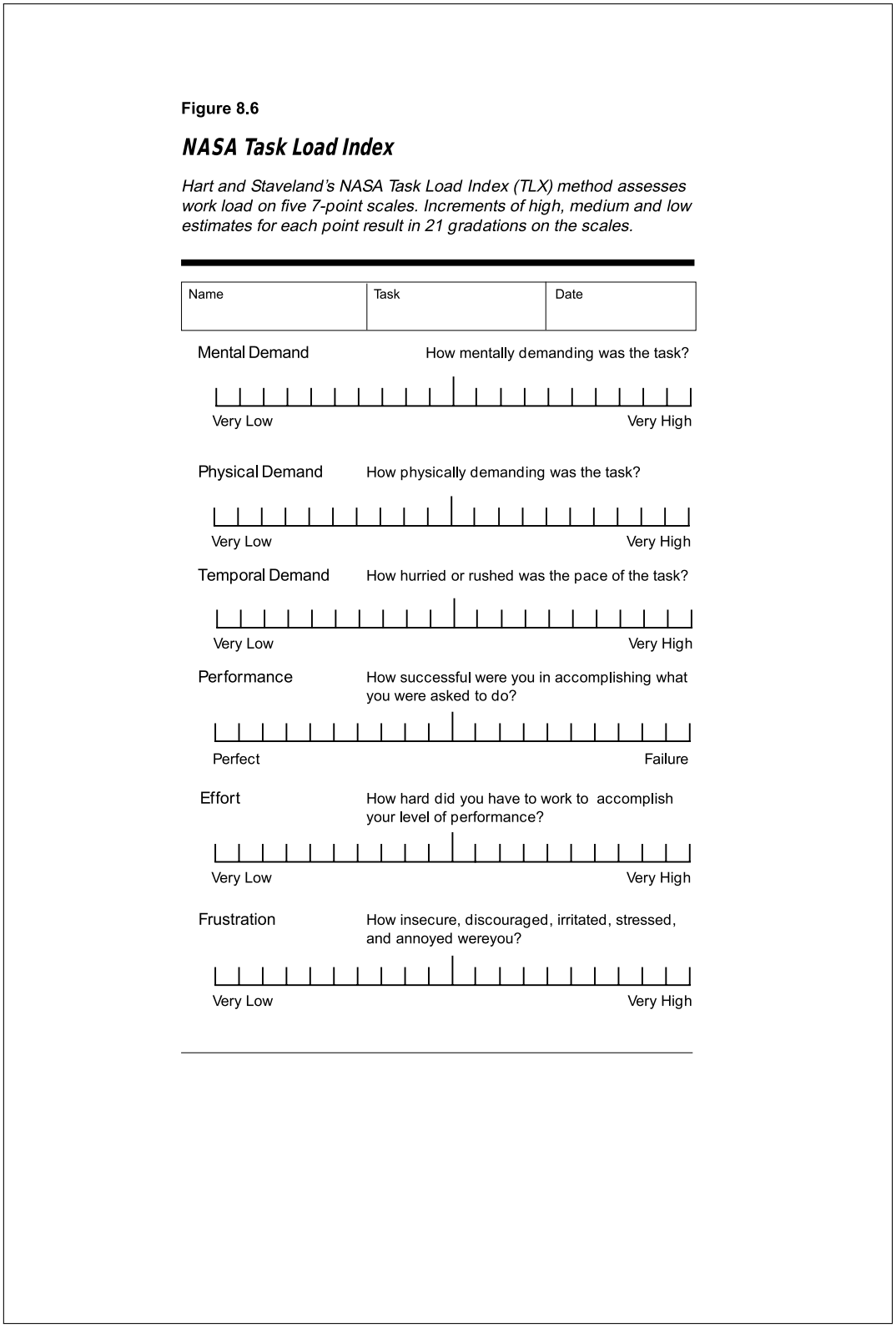
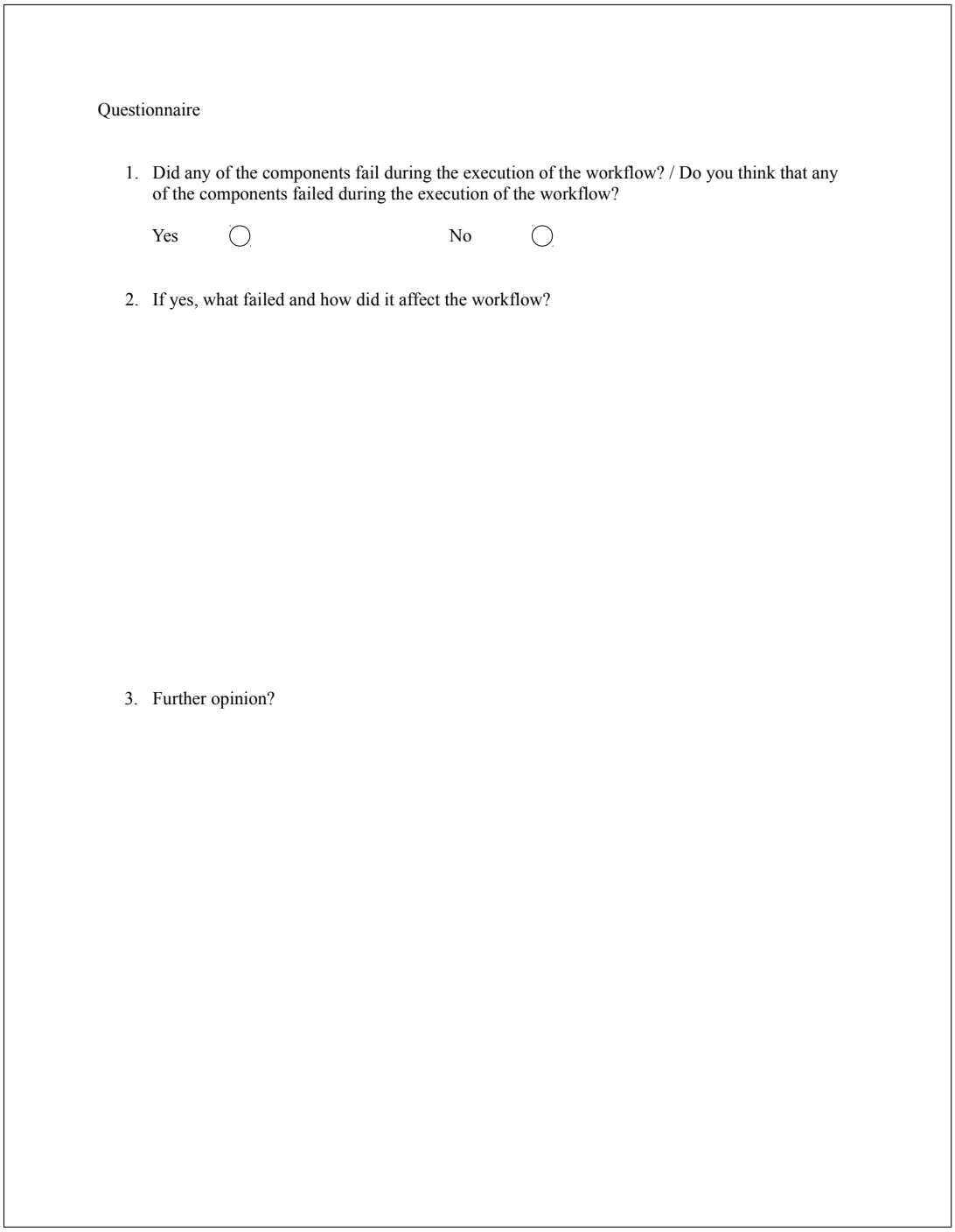


Figure B.3.: Pen/Pencil version of the NASA TLX. Source: [5], ©[2015] NASA



Questionnaire

1. Did any of the components fail during the execution of the workflow? / Do you think that any of the components failed during the execution of the workflow?

   Yes ◯ No ◯

2. If yes, what failed and how did it affect the workflow?

3. Further opinion?

Figure B.4.: The custom questionnaire as used for the evaluation of the workflow-based controller.

Glossary

BPEL Business process execution languages.
BPM Business Process Model.
BPMN Business Process Model and Notation.

CAD Computer Aided Design.
CAS Computer assisted surgery.

DoF Degree of Freedom.
DOR Digital Operating Room.

ERP Enterprise Research Planning.

gSPM generalized Surgical Process Model.

HMI Human Machine Interface.
HMM Hidden Markov Model.

ICP Iterative Closest Point.
IR Infrared.
iSPM individual Surgical Process Model.

NOTES Natural Orifice Transluminal Endoscopic Surgery.

OR Operating Room.
OROCOS Open Robot Control Software.

PCL Point Cloud Library.

ROS Robot Operating System.
ROS-I Robot Operating System Industrial.

SOA Service Oriented Architecture.
SPM Surgical Process Model.
SSH Secure Shell.

TCP Transmission Control Protocol.
ToF Time of Flight.

UDP User Datagram Protocol.
UML Universal Markup Language.

VF Virtual Fixture.

W3C World Wide Web Consortium.
WAPI Workflow Application Programming Interface.
WFM Workflow Management.
WfMC Workflow Management Coalition.

XML Extensible Markup Language.

List of Figures

List of Tables

Bibliography

[1] Owl web ontology language semantics and abstract syntax. http://www.w3.org/TR/owl-semantics/. Last visited: 2015-01-07.

[2] 1000+ ROS packages. http://www.ros.org/news/2010/04/1000-ros-packages.html, 2010. Last visited: 2015-02-13.

[3] realtime_tools. http://wiki.ros.org/realtime_tools, 2013. Last visited: 2015-02-16.

[4] camera calibration package. http://http://wiki.ros.org/camera_calibration, 2015. Last visited: 2015-07-17.

[5] NASA-TLX Paper/Pencil version. http://humansystems.arc.nasa.gov/groups/tlx/paperpencil.html, 2015. Last visited: 2015-08-05.

[6] ROS-industrial. http://rosindustrial.org, 2015. Last visited: 2015-02-16.

[7] UEQ-online. http://www.ueq-online.org, 2015. Last visited: 2015-08-05.

[8] Workflow management coalition. http://www.wfmc.org, 2015. Last visited: 2015-02-16.

[9] World wide web consortium (W3C). http://www.w3.org, 2015. Last visited: 2015-02-16.

[10] M. Adams, A. H. M. Ter Hofstede, and M. La Rosa. Open source software for workflow management: The case of YAWL. *Software, IEEE*, 28(3):16–19, May 2011.

[11] E. H Adelson and J. Y. A. Wang. Single lens stereo with a plenoptic camera. *IEEE transactions on pattern analysis and machine intelligence*, 14(2):99–106, 1992.

[12] S. A. Ahmadi, T. Sielhorst, R. Stauder, M. Horn, H. Feußner, and N. Navab. Recovery of surgical workflow without explicit models. *Medical Image Computing and Computer-Assisted Intervention*, 4190:420–428, 2006.

[13] D. S. Alexiadis, D. Zarpalas, and P. Daras. Real-time, full 3-d reconstruction of moving foreground objects from multiple consumer depth cameras. *IEEE Transactions on Multimedia*, 15(2):339–358, Feb 2013.

[14] G. A. Antoniou, C. V. Riga, E. K. Mayer, N. J. W. Cheshire, and C. D. Bicknell. Clinical applications of robotic technology in vascular and endovascular surgery. *Journal of Vascular Surgery*, 53(2):493–499, 2011.

[15] O. Arsene, I. Dumitrache, and I. Mihu. Expert system for medicine diagnosis using software agents. *Expert Systems with Applications*, 42(4):1825–1834, 2015.

[16] G. Azkune, P. Orduna, X. Laiseca, E. Castillejo, D. Lopez-de Ipina, M. Loitxate, and J. Azpiazu. Semantic framework for social robot self-configuration. *Sensors (Basel)*, 13(6):7004–7020, 2013.

[17] F. Baader, D. Calvanese, D. L. McGuiness, D. Nardi, and P. F. Patel Schneider. The description logic handbook, theory, implementation, and applications (2nd edition). Cambridge University Press, 2010.

[18] R. A. Beasley. Medical robots: Current systems and research directions. *Journal of Robotics*, 2012:1–14, 2012.

[19] J. L. Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, September 1975.

[20] P. J. Besl and N. D. McKay. A method for registration of 3-d shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(2):239–256, February 1992.

[21] R. Bharathan, R. Aggarwal, and A. Darzi. Operating room of the future. *Best Practice and Research: Clinical Obstetrics and Gynaecology*, 27(3):311–322, 2013.

[22] R. Bischoff, J. Kurth, G. Schreiber, R. Koeppe, A. Albu-Schäffer, A. Beyer, O. Eiberger, S. Haddadin, A. S. Stemmer, G. Grunwald, and G. Hirzinger. The KUKA-DLR lightweight robot arm - a new reference platform for robotics research and manufacturing. In *Proceedings of the 41st International Symposium on Robotics (ISR) and 6th German Conference on Robotics (ROBOTIK)*, pages 1–8, 2010.

[23] T. Blum, N. Padoy, H. Feußner, and N. Navab. Workflow mining for visualization and analysis of surgeries. *International journal of computer assisted radiology and surgery*, 3(5):379–386, 2008.

[24] J. Bodner, R. Kafka-Ritsch, P. Lucciarini, J. H. Fish III, and T. Schmid. A critical comparison of robotic versus conventional laparoscopic splenectomies. *World Journal of Surgery*, 29(8):982–985, 2005.

[25] J. Bohren, R. B. Rusu, E. G. Jones, E. Marder-Eppstein, C. Pantofaru, M. Wise, L. Mosenlechner, W. Meeussen, and S. Holzer. Towards autonomous robotic butlers: Lessons learned with the PR2. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5568–5575, May 2011.

[26] L. Bouarfa, P. P. Jonker, and J. Dankelman. Discovery of high-level tasks in the operating room. *Journal of Biomedical Informatics*, 44(3):455–462, 2011.

[27] G. Bradski. OpenCV. *Dr. Dobb's Journal of Software Tools*, 2000.

[28] H. Bruyninckx. Open robot control software: the OROCOS project. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 2523–2528, 2001.

[29] S. E. Butner and M. Ghodoussi. A real-time system for tele-surgery. In *Proceedings of the 21st International Conference on Distributed Computing Systems*, pages 236–243, Apr 2001.

[30] K. Buys, C. Cagniart, A. Baksheev, T. De Laet, J. De Schutter, and C. Pantofaru. An adaptable system for RGB-D based human body detection and pose estimation. *Journal of Visual Communication and Image Representation*, 25(1):39–52, 2014.

[31] M. Caon, Y. Yue, J. Tscherrig, E. Mugellini, and O. Abou Khaled. Context-aware 3D gesture interaction based on multiple kinects. In *Proceedings of the AMBIENT The First International Conference on Ambient Computing, Applications, Services and Technologies*, pages 7–12, 2011.

[32] M. Caversaccio and W. Freysinger. Computer assistance for intraoperative navigation in ENT surgery. *Minimally Invasive Therapy & Allied Technologies*, 12(1-2):36–51, 2003.

[33] M. Chinosi and A. Trombetta. BPMN: An introduction to the standard. *Computer Standards & Interfaces*, 34(1):124–134, 2012.

[34] D. Chrysostomou, N. Kyriakoulis, and A. Gasteratos. Multi-camera 3D scene reconstruction from vanishing points. In *Proceedings of the IEEE International Conference on Imaging Systems and Techniques (IST)*, pages 343–348, July 2010.

[35] K. Cleary, A. Kinsella, and S. K. Mun. OR 2020 workshop report: Operating room of the future. *International Congress Series*, 1281:832–838, 2005.

[36] R. Cucchiara, C. Grana, A. Prati, G. Tardini, and R. Vezzani. Using computer vision techniques for dangerous situation detection in domotic applications. In *Proceedings of Intelligent Distributed Surveillance Systems, IEE*, pages 1–5, Feb 2004.

[37] V. Curcin and M. Ghanem. Scientific workflow systems - can one size fit all? In *Proceedings of the Cairo International Biomedical Engineering Conference (CIBEC)*, pages 1–9, Dec 2008.

[38] A. De Donno, L. Zorn, P. Zanne, F. Nageotte, and M. de Mathelin. Introducing STRAS: A new flexible robotic system for minimally invasive surgery. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1213–1220, May 2013.

[39] Applied Dexterity. Raven. `http://applieddexterity.com/category/raven/`. Last visited: 2015-09-03.

[40] F. Faion, S. Friedberger, A. Zea, and U. D. Hanebeck. Intelligent sensor-scheduling for multi-kinect-tracking. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3993–3999, Oct 2012.

[41] M. A. Fischler and R. C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commununications of the ACM*, 24(6):381–395, June 1981.

[42] S. Franke, P. Liebmann, and T. Neumuth. Connecting workflow management to the or network: Design and evaluation of a bridge to enable dynamic systems behaviour. *Biomedizinische Technik*, 57(SUPPLEMENT 1 TRACK-N):771–774, 2012.

[43] S. B. Gokturk, H. Yalcin, and C. Bamji. A time-of-flight depth sensor - system description, issues and solutions. In *Proceedings of the Conference on Computer Vision and Pattern Recognition Workshop*, pages 35–35, June 2004.

[44] P. Grossmann. Depth from focus. *Pattern Recognition Letters*, 5(1):63–69, 1987.

[45] G. S. Guthart and Jr. Salisbury, J. The Intuitive™ telesurgery system: overview and application. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 618–621 vol.1, 2000.

[46] C. N. Gutt, T. Oniu, A. Mehrabi, A. Kashfi, P. Schemmer, and M. W. Bchler. Robot-assisted abdominal surgery. *British Journal of Surgery*, 91(11):1390–1397, 2004.

[47] U. Hagn, T. Ortmaier, R. Konietschke, B. Kubler, U. Seibold, A. Tobergte, M. Nickl, S. Jorg, and G. Hirzinger. Telemanipulator for remote minimally invasive surgery. *Robotics Automation Magazine, IEEE*, 15(4):28–38, Dec 2008.

[48] T. Haidegger, M. Barreto, P.J.S. Goncalves, M.K. Habib, S.V. Ragavan, H. Li, A. Vaccarella, R. Perrone, and E. Prestes. Robot ontologies for sensor- and image-guided surgery. In *Proceedings of the IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, pages 19–24, 2013.

[49] A. Hampapur, L. Brown, J. Connell, A. Ekin, N. Haas, M. Lu, and S. Pankanti. Smart video surveillance: exploring the concept of multiscale spatiotemporal tracking. *Signal Processing Magazine, IEEE*, 22(2):38–51, March 2005.

[50] B. Hannaford, J. Rosen, D. W. Friedman, H. King, P. Roan, L. Cheng, D. Glozman, J. Ma, S. N. Kosari, and L. White. Raven-II: An open platform for surgical robotics research. *IEEE Transactions on Biomedical Engineering*, 60(4):954–959, 2013.

[51] S. G. Hart and L. E. Staveland. Development of the NASA task load index (TLX): Results of empirical and theoretical research. *P. A. Hancock and N. Meshkati (Eds.), Human Mental Workload*, pages pp. 239–250, 1988.

[52] R. I. Hartley and P. Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

[53] Y.W. Hen and R. Paramesran. Single camera 3d human pose estimation: A review of current techniques. In *Proceedings of the International Conference for Technical Postgraduates (TECHPOS)*, 2009.

[54] D. Herrera C., J. Kannala, and J. Heikkilä. Accurate and practical calibration of a depth and color camera pair. In *Computer Analysis of Images and Patterns*, volume 6855 of *Lecture Notes in Computer Science*, pages 437–445. Springer Berlin Heidelberg, 2011.

[55] M. S. Holden, T. Ungi, D. Sargent, R. C. McGraw, E. C. S. Chen, S. Ganapathy, T. M. Peters, and G. Fichtinger. Feasibility of real-time workflow segmentation for tracked needle interventions. *IEEE Transactions on Biomedical Engineering*, 61(6):1720–1728, June 2014.

[56] D. Hollingsworth. Workflow management coalition specification: The workflow reference model. (TC-1003, 19-jan-95, 1.1). `http://www.wfmc.org`, 1995. Last visited: 2015-09-03.

[57] J. Hutzl, A. Bihlmaier, O. Weede, and H. Wörn. An automated instrument as component of a cognitive medical robotic system for minimal invasive surgery. *Conference of The International Society for Medical Innovation and Technology (iSMIT)*, page 3, 2013.

[58] Intuitive Surgical. *Investor Presentation - Procedures and market opportunitiy*, Q4 2013.

[59] S. Izadi, D. Kim, O. Hilliges, D. Molyneaux, R. Newcombe, P. Kohli, J. Shotton, S. Hodges, D. Freeman, A. Davison, and A. Fitzgibbon. KinectFusion: Real-time 3D reconstruction and interaction using a moving depth camera. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 559–568, 2011.

[60] S. Joyeux, J. Schwendner, and T. M. Roehr. Modular software for an autonomous space rover. In *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS 2014)*. o.A., 6 2014.

[61] C. Kang, D. Kim, W. Lee, and H. Chi. Conventional laparoscopic and robot-assisted spleen-preserving pancreatectomy: does da vinci have clinical advantages? *Surgical Endoscopy*, 25(6):2004–2009, 2011.

[62] B. Karan. Calibration of kinect-type RGB-D sensors for robotic applications. *FME Transactions*, 47:47–54, 2015.

[63] Y. Kassahun, R. Perrone, E. De Momi, E. Berghöfer, L. Tassi, M. P. Canevini, R. Spreafico, G. Ferrigno, and F. Kirchner. Automatic classification of epilepsy types using ontology-based and genetics-based machine learning. *Artificial Intelligence in Medicine*, 61(2):79–88, 2014.

[64] D. Katić, A.-L. Wekerle, F. Gärtner, H. Kenngott, B. P. Müller-Stich, R. Dillmann, and S. Speidel. Knowledge-driven formalization of laparoscopic surgeries for rule-based intraoperative context-aware assistance. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8498 LNCS:158–167, 2014.

[65] E. R. Kim, C. Lim, D. J. Kim, J. S. Kim, and K. H. Park. Robot-assisted cardiac surgery using the da vinci surgical system: a single center experience. *The Korean Journal of Thoracic and Cardiovascular Surgery*, 48(2):99–104, Apr 2015.

[66] J. H. Kim, Choim J. S., and B. K. Koo. Calibration of multi-kinect and multi-camera setup for full 3d reconstruction. In *Proceedings of the 44th International Symposium on Robotics (ISR)*, pages 1–5, Oct 2013.

[67] S. J. Kim, M. MacDonald, J. Hernandez, and R. L. Wixson. Computer assisted navigation in total knee arthroplasty: improved coronal alignment. *The Journal of arthroplasty*, 20:123–131, 2005.

[68] R. Konietschke, U. Hagn, M. Nickl, S. Jorg, A. Tobergte, G. Passig, U. Seibold, L. Le-Tien, B. Kubler, M. Groger, F. Frohlich, C. Rink, A. Albu-Schäffer, M. Grebenstein, T. Ortmaier, and G. Hirzinger. The DLR MiroSurge - a robotic system for surgery. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1589–1590, 2009.

[69] M. Kranzfelder, C. Staub, A. Fiolka, A. Schneider, S. Gillen, D. Wilhelm, H. Friess, A. Knoll, and H. Feußner. Toward increased autonomy in the surgical OR: Needs, requests, and expectations. *Surgical Endoscopy and Other Interventional Techniques*, 27(5):1681–1688, 2013.

[70] F. Kühn and M. Leucker. OR.NET: Safe interconnection of medical devices. In *Foundations of Health Information Engineering and Systems*, pages 188–198. Springer, 2014.

[71] Y. S. Kwoh, J. Hou, E. A. Jonckheere, and S. Hayati. A robot with improved absolute positioning accuracy for CT guided stereotactic brain surgery. *IEEE Transactions on Biomedical Engineering*, 35(2):153–160, Feb 1988.

[72] A. Ladikos, S. Benhimane, and N. Navab. Real-time 3D reconstruction for collision avoidance in interventional environments. In *Medical Image Computing and Computer-Assisted Intervention-MICCAI 2008*, pages 526–534. Springer, 2008.

[73] F. Lalys and P. Jannin. Surgical process modelling: A review. *International Journal of Computer Assisted Radiology and Surgery*, 9(3):495–511, 2014.

[74] M. J. Lang, A. D. Greer, and G. R. Sutherland. Intra-operative robotics: Neuroarm. *Acta neurochirurgica. Supplement*, 109:231–236, 2011.

[75] K. B. Laskey. MEBN: A language for first-order bayesian knowledge bases. *Artificial Intelligence*, 172(2-3):140–178, February 2008.

[76] B. Laugwitz, T. Held, and M. Schrepp. Construction and evaluation of a user experience questionnaire. In *Proceedings of the 4th Symposium of the Workgroup Human-Computer Interaction and Usability Engineering of the Austrian Computer Society on HCI and Usability for Education and Work*. Springer, 2008.

[77] H. U. Lemke and L. Berliner. The digital operating room: Towards intelligent infrastructures and processes. In *SPIE Medical Imaging*, volume 8674, 2013.

[78] S. Li, P. N. Pathirana, and T. Caelli. Multi-kinect skeleton fusion for physical rehabilitation monitoring. In *Proceedings of the 36th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, pages 5060–5063, Aug 2014.

[79] P. Liebmann, J. Meixensberger, P. Wiedemann, and T. Neumuth. The impact of missing sensor information on surgical workflow management. *International Journal of Computer Assisted Radiology and Surgery*, 8(5):867–875, 2013.

[80] P. Liebmann and T. Neumuth. Model driven design of workflow schemata for the operating room of the future. In *Proceedings of the GI Jahrestagung (1)*, volume 1, pages 415–419, 2010.

[81] R. Macknojia, A. Chavez-Aragon, P. Payeur, and R. Laganiere. Calibration of a network of kinect sensors for robotic inspection over a large workspace. In *Proceedings of the IEEE Workshop on Robot Vision (WORV)*, pages 184–190, Jan 2013.

[82] D. W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 1963.

[83] D. L. McGuinness and F. van Harmelen. OWL web ontology language overview. Technical Report REC-owl-features-20040210, W3C, 2004.

[84] Ph. Merloz, J. Tonetti, L. Pittet, M. Coulomb, S. Lavallee, J. Troccaz, P. Cinquin, and P. Sautot. Computer-assisted spine surgery. *Computer Aided Surgery*, 3(6):297–305, 1998.

[85] N. Mohamed, J. Al-Jaroodi, and I. Jawhar. Middleware for robotics: A survey. In *Proceedings of the IEEE Conference on Robotics, Automation and Mechatronics*, pages 736–742, Sept 2008.

[86] H. Mönnich. *Ein Steuersystem fuer die telemanipulierte und autonome robotergestuetzte Chirurgie (German Edition)*. KIT Scientific Publishing, 2012.

[87] B. Motik, R. Shearer, and I. Horrocks. Hypertableau reasoning for description logics. *Journal of Artificial Intelligence Research*, 36(1):165–228, 2009.

[88] J. Münchenberg, J. Brief, J. Raczkowsky, H. Wörn, S. Haßfeld, and J. Mühling. Operation planning of robot supported surgical interventions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 1, pages 547–552, 2000.

[89] J. E. Münchenberg. *Rechnergestützte Operationsplanung in der Mund-Kiefer-Gesichts-Chirurgie*. Forschen und Wissen - Informatik. GCA, Herdecke, 2001.

[90] M. Nakazawa, I. Mitsugami, Y. Makihara, H. Nakajima, H. Habe, H. Yamazoe, and Y. Yagi. Dynamic scene reconstruction using asynchronous multiple kinects. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR)*, pages 469–472, 2012.

[91] D. Neumuth, F. Loebe, H. Herre, and T. Neumuth. Modeling surgical processes: A four-level translational approach. *Artificial Intelligence in Medicine*, 51(3):147–161, 2011.

[92] T. Neumuth, A. Krauss, J. Meixensberger, and O. J. Muensterer. Impact quantification of the davinci telemanipulator system on surgical workflow using resource impact profiles. *International Journal of Medical Robotics and Computer Assisted Surgery*, 7(2):156–164, 2011.

[93] T. Neumuth, C. Trantakis, F. Eckhardt, M. Dengl, J. Meixensberger, and O. Burgert. Supporting the analysis of intervention courses with surgical process models on the example of fourteen microsurgical lumbar discectomies. *International Journal of Computer Assisted Radiology and Surgery*, 2(1):436–438, 2007.

[94] P. Nicolai, T. Beyl, H. Mönnich, J. Raczkowsky, and H. Wörn. Op:sense an integrated rapid development environment in the context of robot assisted surgery and operation room sensing. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 2421–2422, 2011.

[95] P. Nicolai, J. Raczkowsky, and H. Wörn. A novel 3D based camera system for safe human-robot interaction in the operating room. *Journal of Automation and Control Engineering*, pages 410–417, 2014.

[96] L. W. Nifong, W. R. Chitwood, P. S. Pappas, C. R. Smith, M. Argenziano, V. A. Starnes, and P. M. Shah. Robotic mitral valve surgery: a united states multicenter trial. *Journal of Thoracic and Cardiovascular Surgery*, 129(6):1395–1404, Jun 2005.

[97] S. Obdrzalek, G. Kurillo, F. Ofli, R. Bajcsy, E. Seto, H. Jimison, and M. Pavel. Accuracy and robustness of kinect pose estimation in the context of coaching of elderly population. In *Proceedings of the Annual*

International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), pages 1188–1193, Aug 2012.

[98] N. Padoy, T. Blum, S.-A. Ahmadi, H. Feußner, M.-O. Berger, and N. Navab. Statistical modeling and recognition of surgical workflow. *Medical Image Analysis*, 16(3):632–641, 2012.

[99] N. Padoy, T. Blum, H. Feußner, M.-O. Berger, and N. Navab. On-line recognition of surgical activity for monitoring in the operating room. In *Proceedings of the 20th National Conference on Innovative Applications of Artificial Intelligence*, volume 3 of *IAAI'08*, pages 1718–1724. AAAI Press, 2008.

[100] N. Padoy, D. Mateus, D. Weinland, M.-O. Berger, and N. Navab. Workflow monitoring based on 3d motion features. In *Proceedings of the IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 585–592, Sept 2009.

[101] V. R. Patel, M. F. Chammas Jr., and S. Shah. Robotic assisted laparoscopic radical prostatectomy: a review of the current state of affairs. *International Journal of Clinical Practice*, 61(2):309–314, Feb 2007.

[102] H. A. Paul, W. L. Bargar, B. Mittlestadt, P. Kazanzides, B. Musits, J. Zuhars, P. W. Cain, B. Williamson, and F. G. Smith. Robotic execution of a surgical plan. In *IEEE International Conference on Systems, Man and Cybernetics*, pages 1621–1623 vol.2, Oct 1992.

[103] A. Payne, A. Daniel, A. Mehta, B. Thompson, C. S. Bamji, D. Snow, H. Oshima, L. Prather, M. Fenton, L. Kordus, P. O'Connor, R. McCauley, S. Nayak, S. Acharya, S. Mehta, T. Elkhatib, T. Meyer, T. O'Dwyer, T. Perry, Vei-Han Chan, V. Wong, V. Mogallapu, W. Qian, and Z. Xu. A 512x424 CMOS 3D Time-of-Flight image sensor with multi-frequency photo-demodulation up to 130MHz and 2GS/s ADC. In *Proceedings of the IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 134–135, Feb 2014.

[104] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, 2009.

[105] C. Raposo, J. P. Barreto, and U. Nunes. Fast and accurate calibration of a kinect sensor. In *Proceedings of the International Conference on 3D Vision (3DV 2013)*, pages 342–349, June 2013.

[106] P. N. Robinson and S. Bauer. *Introduction to Bio-Ontologies*. Chapman & Hall/CRC Mathematical and Computational Biology. CRC Press, 2011.

[107] M. Ruffolo, M. Manna, V. Cozza, and R. Ursino. Semantic clinical process management. In *Proceedings fo the Twentieth IEEE International Symposium on Computer-Based Medical Systems (CBMS)*, pages 518–523, June 2007.

[108] R. B. Rusu and S. Cousins. 3D is here: Point cloud library (PCL). In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[109] A. Sabov and J. Krüger. Identification and correction of flying pixels in range camera data. In *Proceedings of the 24th Spring Conference on Computer Graphics*, SCCG '08, pages 135–142, New York, NY, USA, 2010. ACM.

[110] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. In *Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 1, pages I–195–I–202 vol.1, June 2003.

[111] O. Schorr. *Operationsplanung und -steuerung in der Chirurgie*. Logos, 2005.

[112] L. Schreiter, E. Berghöfer, T. Beyl, L. Senger, J. Raczkowsky, F. Kirchner, and H. Wörn. Probabilistic situation detection for human-robot interaction in the operation theatre. *Tagungsband der 14. Jahrestagung der Deutschen Gesellschaft für Computer- und Roboterassistierte Chirurgie e.V*, 2015.

[113] R. Shearer, B. Motik, and I. Horrocks. HermiT: A highly-efficient owl reasoner. In *Proceedings of the 5th International Workshop on OWL: Experiences and Directions (OWLED 2008 EU)*, Karlsruhe, Germany, October 26-27 2008.

[114] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, and R. Moore. Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124, 2013.

[115] A. Shpunt and Z. Zalevsky. Depth-varying light fields for three dimensional sensing, 2008. US Patent App. 11/724,068.

[116] A. Shpunt and Z. Zalevsky. Three-dimensional sensing using speckle patterns, 2009. WO Patent App. PCT/IL2007/000,306.

[117] E. Sirin and B. Parsia. SPARQL-DL: SPARQL query for OWL-DL. In *Proceedings of the 3rd OWL Experiences and Directions Workshop (OWLED 2007)*, 2007.

[118] E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53, 2007.

[119] R. Stauder, A. Okur, L. Peter, A. Schneider, M. Kranzfelder, H. Feußner, and N. Navab. Random forests for phase detection in surgical workflow analysis. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 8498 LNCS:148–157, 2014.

[120] D. Stengel, T. Wiedemann, and B. Vogel-Heuser. Efficient 3D voxel reconstruction of human shape within robotic work cells. In *Proceedings of the International Conference on Mechatronics and Automation (ICMA)*, pages 1386–1392, 2012.

[121] G. Strauss, M. Fischer, J. Meixensberger, V. Falk, C. Trantakis, D. Winkler, F. Bootz, O. Burgert, A. Dietz, and H. U. Lemke. Workflow analysis to assess the efficiency of intraoperative technology using the example of functional endoscopic sinus surgery. *HNO*, 54(7):528, 2006.

[122] T. Sugino, H. Kawahira, and R. Nakamura. Surgical task analysis of simulated laparoscopic cholecystectomy with a navigation system. *International Journal of Computer Assisted Radiology and Surgery*, 9(5):825–836, 2014.

[123] Intuitive surgical. Image Gallery. `http://intuitivesurgical.com/company/media/images`. Last visited: 2015-01-14.

[124] R. H. Taylor. A perspective on medical robotics. *Proceedings of the IEEE*, 94(9):1652–1664, September 2006.

[125] M. Tenorth and M. Beetz. KNOWROB - knowledge processing for autonomous personal robots. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4261–4266, 2009.

[126] J. Tokuda, G. S. Fischer, X. Papademetris, Z. Yaniv, L. Ibanez, P. Cheng, H. Liu, J. Blevins, J. Arata, A. J. Golby, T. Kapur, S. Pieper, E. C. Burdette, G. Fichtinger, C. M. Tempany, and N. Hata. OpenIGTLink: An open network protocol for image-guided therapy environment. *International Journal of Medical Robotics and Computer Assisted Surgery*, 5(4):423–434, 2009.

[127] D. Tsarkov and I. Horrocks. FaCT++ description logic reasoner: System description. In *Proceedings of the Third International Joint Conference on Automated Reasoning*, IJCAR'06, pages 292–297, Berlin, Heidelberg, 2006. Springer-Verlag.

[128] W. M. P. Van der Aalst. Business process management: A comprehensive survey. *ISRN Software Engineering*, 2013:1–37, 2013.

[129] W. M. P. Van der Aalst and A. H. M. Ter Hofstede. YAWL: yet another workflow language. *Information systems*, 30(4):245–275, 2005.

[130] W. M. P. Van der Aalst, B. F. Van Dongen, C. W. Günther, A. Rozinat, E. Verbeek, and T. Weijters. ProM: The process mining toolkit. In *Proceedings of the Business Process Management Demonstration Track (BPMDemos 2009)*, 2009.

[131] T. A. S. C. Vieira, M. A. Casanova, and L. G. Ferrao. An ontology-driven architecture for flexible workflow execution. In *Proceedings of the WebMedia and LA-Web*, pages 70–77, Oct 2004.

[132] P. Štádler, L. Dvořáček, P. Vitásek, and P. Matouš. The application of robotic surgery in vascular medicine. *Innovations: Technology and Techniques in Cardiothoracic and Vascular Surgery*, 7(4):247–253, 2012.

[133] G. J. Wang, D. A. Barocas, J. D. Raman, and D. S. Scherr. Robotic vs open radical cystectomy: Prospective comparison of perioperative outcomes and pathological measures of early oncological efficacy. *BJU International*, 101(1):89–93, 2008.

[134] O. Weede, F. Dittrich, H. Wörn, B. Jensen, A. Knoll, D. Wilhelm, M. Kranzfelder, A. Schneider, and H. Feußner. Workflow analysis and surgical phase recognition in minimally invasive surgery. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1080–1074, Dec 2012.

[135] A. D. Wilson and H. Benko. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pages 273–282, New York, NY, USA, 2010. ACM.

[136] R. S. Yang, Y. H. Chan, R. Gong, M. Nguyen, A. G. Strozzi, P. Delmas, G. Gimel'farb, and R. Ababou. Multi-kinect scene reconstruction: Calibration and depth inconsistencies. In *Proceedings of the 28th International Conference of Image and Vision Computing New Zealand (IVCNZ)*, pages 47–52, Nov 2013.

[137] Y. Yang, F. Wang, P. Zhang, C. Shi, Y. Zou, H. Qin, and Y. Ma. Robot-assisted versus conventional laparoscopic surgery for colorectal disease, focusing on rectal cancer: A meta-analysis. *Annals of Surgical Oncology*, 19(12):3727–3736, 2012.

[138] P. Yohannes, P. Rotariu, P. Pinto, A. D. Smith, and B. R. Lee. Comparison of robotic versus laparoscopic skills: Is there a difference in the learning curve? *Urology*, 60(1):39–45, 2002.

[139] M. K. Young, C. Theobalt, J. Diebel, J. Kosecka, B. Miscusik, and S. Thrun. Multi-view image and ToF sensor fusion for dense 3D reconstruction. In *Proceedings of the IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1542–1549, Sept 2009.

[140] M. Zairi. Business process management: a boundaryless approach to modern competitiveness. *Business Process Management Journal*, 3(1):64–80, 1997.

[141] J. Zhang, X. Lu, H. Duan, and H. Nie. A medical information system architecture based on workflow technology. In *Proceedings of the IEEE International Symposium on IT in Medicine Education (ITIME)*, volume 1, pages 1117–1121, Aug 2009.

[142] L. Zhang, J. Sturm, D. Cremers, and D. Lee. Real-time human motion tracking using multiple depth cameras. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2012.*, pages 2389–2395, 2012.