Human Pose Estimation with Supervoxels

zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften

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

genehmigte

Dissertation

von

Alexander Schick

aus Landau in der Pfalz

Tag der mündlichen Prüfung:	8. Mai 2014
Erster Gutachter.	Prof. Dr.-Ing. Rainer Stiefelhagen
Zweiter Gutachter:	Prof. Dr.-Ing. Tamim Asfour

Zusammenfassung

Intelligente Umgebungen sind in der Lage, Menschen wahrzunehmen und auf ihre Handlungen zu reagieren. Ein Ziel ist es dabei, die Menschen bei ihren Tätigkeiten zu unterstützen. Dafür soll die Interaktion in und mit solchen Umgebungen möglichst natürlich und intuitiv sein, weshalb der Schnittstelle zwischen Mensch und Maschine besondere Bedeutung zukommt. Um die Aktivitäten von Menschen zu verstehen, ist es in einem ersten Schritt notwendig, ihre Bewegungen und insbesondere Gesten zu erkennen. In dieser Arbeit wurde dazu ein System entwickelt, welches die Körperposen von Personen erkennt und dadurch eine gestenbasierte Interaktion in und mit einer intelligenten Umgebung ermöglicht.

Die Forschungsfrage dieser Arbeit lautet dabei: *Wie kann durch Segmentierung als ein Vorverarbeitungsschritt sowohl der Suchraum als auch die Komplexität bei der Erfassung der Körperpose reduziert werden?* Diese Frage wurde für alle Schritte des im Rahmen dieser Arbeit entwickelten Systems untersucht.

Im ersten Verarbeitungsschritt werden die Sensordaten durch speziell angepasste Voxel Carving Verfahren in eine sensorunabhängige Repräsentationsform, die Voxel, umgewandelt. Dies führt zu einer Flexibilität bezüglich der Sensorauswahl und erlaubt dadurch den Einsatz in unterschiedlichen Umgebungen. Der Algorithmus ist dabei sehr effizient und robust gegenüber statischen Verdeckungen durch die Verwendung sogenannter Occlusion Maps. Zusätzlich dazu wurden im Rahmen dieser Arbeit Superpixel- und Supervoxel-Segmentierungsalgorithmen untersucht und entwickelt. Insbesondere die Segmentierung in Supervoxel ermöglicht eine Reduktion des Suchraums für die Algorithmen der folgenden Verarbeitungsschritte.

Für die Erfassung der 3D Körperpose werden Supervoxel und die Verbindungen des Supervoxel-Graphen als Grundbausteine verwendet. Dadurch wird sowohl der Suchraum als auch die Komplexität reduziert und eine Erfassung der Körperpose in Echtzeit ermöglicht. Zudem führt die Verwendung von Supervoxeln zu einer größeren Flexibilität, da zusätzliche Informationen auf Supervoxelbasis modelliert werden können ohne eine Modifikation des Algorithmus zu benötigen. Dies wird am Beispiel der temporalen Supervoxeln gezeigt, mit welchen Informationen über vergangene Posen in die Zukunft propagiert werden können.

Das System zur Erfassung der Körperpose wurde sowohl für die Erkennung von statischen als auch dynamischen Gesten verwendet und seine Eignung in verschiedenen Laboraufbauten, Ausstellungen und industriellen Umgebungen demonstriert.

Zusammenfassend erlaubt das vorgestellte System eine Erfassung der Körperpose für die Mensch-Computer-Interaktion in Echtzeit und erreicht eine signifikante Reduktion sowohl der Komplexität als auch des Suchraums durch die Verwendung von Supervoxel-Segmentierungen.

Abstract

Smart environments provide natural and intuitive human-computer interaction to support people in their daily lives. For this functionality, the computers must be able to perceive people and recognize their actions. In this thesis, a system was developed that enables interactions in and with smart environments by recognizing gestures based on pose estimation and body tracking.

The main research question of this thesis is: *How can segmentation as a preprocessing step reduce the search space and computational complexity of pose estimation and body tracking?* This question has been investigated for all steps of the system presented in this thesis.

The starting point is a segmentation into voxels with voxel carving to compute a sensor-independent representation. This provides flexibility as different sensor types are better suited for various environments. The algorithm is robust to static occlusions through occlusion maps and achieves a high computational efficiency. Further, both superpixel and supervoxel segmentation algorithms were developed. Through supervoxels, the large number of voxels, and therefore the search space, is drastically reduced for following processing steps.

The 3D pose estimation approach uses supervoxels and connections of the supervoxel graph as building blocks. This significantly reduces both the search space as well as the computational complexity and allows for real-time pose estimation. Further, supervoxels provide an additional flexibility because they can encode diverse information to influence pose estimation without modifying the approach. This is shown with temporal pictorial structures that propagate previous poses through time for body tracking.

The system has been applied to static and dynamic gesture recognition and its applicability was demonstrated in various laboratory settings, exhibitions, and industrial environments.

In conclusion, the presented system allows real-time human-computer interaction based on pose estimation and body tracking and achieves a significant computational complexity as well as search space reduction through segmentation techniques.

Acknowledgments

This dissertation was conducted at the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation (IOSB) in Karlsruhe. I would like to express my greatest thanks to my supervisor Prof. Dr.-Ing. Rainer Stiefelhagen. He offered me the opportunity to join his new research group at the Fraunhofer IOSB to work on the fascinating topic of human-computer interaction that I was interested in since the early days of my studies. His guidance and advice kept me on track towards my research goals and his supervision was invaluable for my work. I also thank my co-referee Prof. Dr.-Ing. Tamim Asfour for supporting me throughout various phases of my academic life. The experiences I gained and the support that I received shaped the path of my research.

I thank my closest colleagues Florian van de Camp, Joris Ijsselmuiden, and Dr.-Ing. Michael Voit with whom I formed the SmartControlRoom team. It was an unforgettable time and great pleasure to work with you and I will always look back with joy at our shared experiences. I thank all the people, past and present, of the department of Interactive Analysis and Diagnosis at the Fraunhofer IOSB. It is the perfect place to conduct applied research in a warm and friendly environment. Many thanks go to my colleagues at the Computer Vision for Human-Computer Interaction Lab at the Karlsruhe Institute of Technology for fruitful discussions and joint research projects. Often, it was your comments that showed me the next step in my research. I want to express my great thanks to Prof. Dr.-Ing. Jürgen Beyerer and Dr. Geisler for bringing this research project into existence and to Dr. Elisabeth Peinsipp-Byma for her continuing support.

I thank Martin Bäuml, Jürgen Brauer, Mika Fischer, and Dr. Justyna Gansel for proofreading. Your valuable comments and suggestions helped to improve this document and contributed towards its current form.

My deepest thanks go to my parents Hans and Gabriele Schick. You enabled and encouraged me to follow the path of my choosing and I will be forever thankful for your never ending support. I thank my fiancée Justyna Gansel for traveling this challenging path together with me. No matter how difficult the times, I could always count on your love and support.

Contents

List of Figures

List of Tables

1 Introduction

This chapter introduces the topics and research challenges addressed in this thesis. First, a motivation is given in Section 1.1. Then, an overview of the approach is presented in Section 1.2. Contributions are summarized in Section 1.3 and an outline is presented in Section 1.4.

1.1 Motivation

This thesis took place during a thriving time as the interaction between humans and computers has been undergoing drastic change over the last years. Interaction shifted from mouse and keyboards to natural interaction with gestures, in particular in combination with various display types. What a couple of years ago seemed futuristic, is now available for mass markets. Smart phones are now an integral part of our daily lives and gesture-controlled interaction in millions of living rooms is no concept of the future anymore.

Smart rooms address a specific topic of human-computer interaction. The research in this area focuses on how whole rooms can be made *smarter* with the goal to support humans in their daily activities. Fundamental research about *Computers in the Human Interaction Loop* [169] at the Karlsruhe Insitute of Technology (formerly known as Universität Karlsruhe (TH)) provided the seeds for smart room research in Karlsruhe. Enabled by this work, the new research group *Visual Perception for Human-Computer Interaction - Interaction in and with Smart Environments* and the *SmartControlRoom* [10] started at the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation in Karlsruhe. In this research environment, this thesis is founded.

To understand humans and their interactions, computers must be able to recognize their actions. By measuring the poses of people and tracking them over time, the basic information required for understanding human interactions is provided. This thesis explores an approach to estimate and track the poses to provide gesture-based interaction for smart environments.

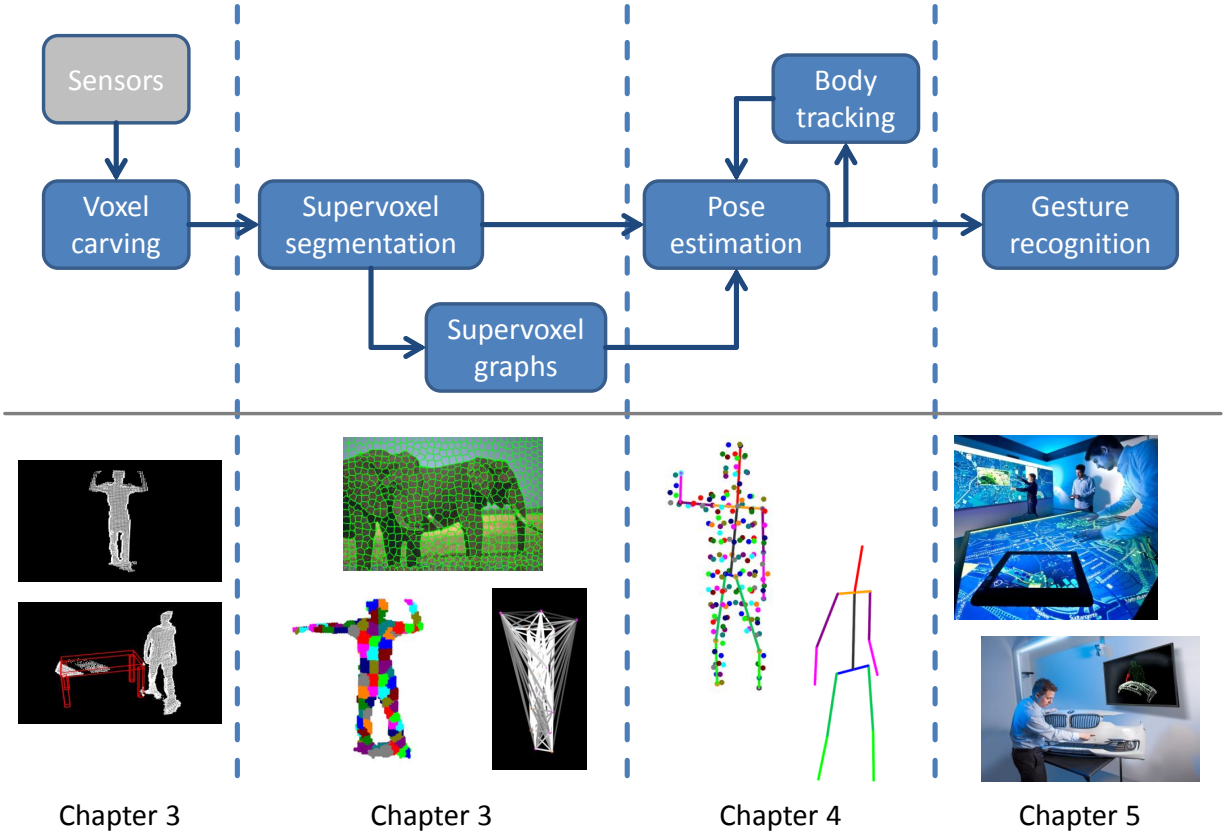


Figure 1.1: System overview.

1.2 Approach

One goal of this thesis was the development of a system for real-time human-computer interaction with gestures. Therefore, the whole processing pipeline starting from sensor data acquisition to gesture-based interaction was investigated and addressed in this work. Figure 1.1 shows an overview of the system.

For human-computer interaction, both robustness and flexibility are desirable properties. Through voxel carving as the first processing step, a sensor-independent 3D reconstruction is computed. This approach has the advantage that it supports various sensor types for improved flexibility.

The resulting voxel reconstruction contains a large number of voxels as input for the following processing steps. This directly influences the overall runtime. Therefore, super segmentation methods into superpixels and supervoxels have been investigated. In particular the segmentation of voxels into supervoxels and the construction of the supervoxel graph is a fundamental element to achieve a reduction of the search space for pose estimation.

For 3D pose estimation, supervoxels are applied as building blocks to pictorial structures. Through this step, both the search space as well as the overall complexity is drastically reduced and poses can be directly estimated with little prior knowledge. Further, additional temporal information can directly be integrated for articulated body tracking.

Based on these processing steps, a gesture-based interaction system was developed that allows for both static as well as dynamic gestures for intuitive and natural human-computer interaction.

1.3 Contributions

This thesis addresses challenges in four different areas of computer vision and contributes towards solutions. The research areas are (1) volumetric reconstruction, (2) image and volume segmentation, (3) pose estimation and body tracking, and (4) interaction application. The main research question that unifies these contributions is:

How can segmentation as a preprocessing step reduce the search space and computational complexity of pose estimation and body tracking?

For volumetric reconstruction, an efficient voxel carving algorithm was developed. It is faster than comparable approaches [91] and offers robustness against static occluders by incorporating occlusion maps. Further, it supports various sensor types by providing different carving functions. In this work, it has been applied to multi-view video and both single and multi-view depth sensors. As different environments can require different sensor types for optimal operation, this enhances the overall flexibility. Even more so because this flexibility is transferred over to applications working with these voxels.

Superpixel segmentation is a segmentation technique that partitions an image into compact and homogeneous regions, the superpixels [181]. The main advantage of superpixels is that the number of inputs for following algorithms is reduced because similar pixels are grouped to a single unit, or primitive. In this work, a superpixel segmentation algorithm was developed based on [19] that works on boundary evolution and that outperforms the current state-of-the-art. Further, the concept of superpixel compactness was investigated and a compactness metric developed and introduced.

The 2D superpixel segmentation was further improved to also compute 3D supervoxel segmentation of volumetric data represented by voxels. In addition, a 3D supervoxel compactness metric was presented. The algorithm was compared to the currently only

existing other supervoxel approach for 3D voxels [45] and it was shown that it achieves better results for segmentation accuracy and undersegmentation error with similar compactness. In addition, it is computationally more efficient.

In recent years, part detectors have become popular both for 2D [38, 39] and 3D pose estimation [146]. In particular, 3D pose estimation can be considered solved for depth sensors and huge amounts of training data [146]. Part detectors steer pose estimation towards poses that are in accordance with the locations of detected body parts. This can be used to make pose estimation more accurate and efficient, but also introduces a bias through the training data. Therefore, this work follows a different approach. It investigates how the search space of possible poses can be reduced through segmentation as a preprocessing step to directly estimate the best pose. In particular, it does not rely on part detectors that can potentially be biased towards the observed appearances. It shows how pictorial structures can be efficiently used for 3D pose estimation by using supervoxels as primitives. The resulting search space and complexity reduction allows for very short computation times that enable real-time human-computer interaction. By using supervoxels to reduce the search space, poses can be directly estimated with only little prior knowledge. Further, the concept of supervoxel energies is presented that allows integration of additional information. In this work, supervoxel energies are used to propagate information through time for articulated body tracking.

Based on this work, a gesture recognition system was developed that recognizes both static touch and pointing gestures as well as dynamic gestures. It allows for a natural and intuitive human-computer interaction. The robustness of this system has been tested in both laboratory and real-world settings, including various exhibitions and industrial applications.

1.4 Outline

This work is organized as follows. Related work is discussed in Chapter 2 with a focus on volumetric reconstruction, super segmentation techniques, and pose estimation and articulated body tracking.

Chapter 3 presents segmentation techniques developed in this work. This includes a voxel carving algorithm for 3D reconstruction, superpixel and supervoxel segmentation algorithms for a reduction of the number of input elements, and supervoxel graphs to model connections between supervoxels.

Chapter 4 introduces the pose estimation approach and articulated body tracking. It achieves a computational complexity reduction by using supervoxels as primitives and requires very little prior information.

Chapter 5 describes the gesture recognition system that was developed in the context of this work. Through gesture recognition for human-computer interaction, people can interact with both static as well as dynamic gestures. The system was used in laboratory setups, during exhibitions, and in industrial environments.

This work concludes with a discussion in Chapter 6 and an outlook to future work.

2 Background and Related Work

This chapter presents an overview of related work. As the introduction in Chapter 1 showed, this work contributed to four areas of computer vision, namely (1) volumetric reconstruction, (2) image segmentation, (3) pose estimation and body tracking, and (4) gesture interaction for human-computer interaction. As the main topic of this work is on pose estimation with a focus on search space and complexity reduction through segmentation, the discussion of related work will focus on the involved topics.

First, methods for volumetric reconstruction with multiple cameras are described in Section 2.1. Then, super segmentation methods, *i.e.*, superpixel and supervoxel segmentations that reduce size of the input data, are discussed in Section 2.2. The chapter concludes with an overview of pose estimation and body tracking approaches in Section 2.3.

2.1 Volumetric Reconstruction

This section describes methods to obtain an approximation of the 3D shapes of objects observed by multiple video cameras. In computer vision, these methods are generally grouped under the name *shape-from-X* where X specifies the image feature used for approximation. Examples include shape-from-motion [156, 159], shape-from-shading [79, 184], shape-from-focus [117], shape-from-texture [25], or shape-from-silhouette [103]. In this work, shape-from-silhouette is considered more closely because it is well-suited for static multi-camera setups that are commonly used in smart environments. Before introducing shape-from-silhouette methods in Section 2.1.3, the next two sections give an overview of camera calibration and foreground segmentation methods that are common requirements for 3D reconstruction.

2.1.1 Camera Calibration

A camera gives an image of the observed world. The transformations between the 2D camera image and the 3D world are given by projective geometry. With these equations,

3D points can be projected into the camera image, image points can be transformed back to lines in 3D, and points can be converted between coordinate systems of different cameras. To solve these equations, the camera parameters are required.

The camera parameters are determined by a process called camera calibration. This work uses the pinhole camera as reference model and the camera is fully described with the *intrinsic* parameters, that describe the camera internals, *i.e.*, the lens, and the *extrinsic* parameters that describe the camera position relative to a global world coordinate system. With both intrinsic and extrinsic parameters, transformations between the world and camera coordinate systems are possible as well as projections from the camera coordinate system into image space.

The next sections first introduce the required equations to project 3D points into an image and to convert 3D points between different coordinate systems, thereby also exemplifying the functions of both intrinsic and extrinsic parameters. Then, the camera calibration is described for both video cameras and depth sensors.

Intrinsic Parameters for 3D-to-2D Projections

The intrinsic parameters describe the camera internals and are required to project points from the 3D camera coordinate system into its image. They include the focal lengths f_x and f_y given in pixels, the principal point (c_x, c_y), and the skewness γ of the image axes. In practice, the skew factor can often be neglected because the pixels are almost perfectly rectangular with a skew factor close to zero. The intrinsic parameters form the camera matrix K:

$$K = \begin{pmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}. \tag{2.1}$$

Let $p = (x, y, z)$ be a 3D point in the camera coordinate system with $x' = \frac{x}{z}$ and $y' = \frac{y}{z}$. The projection to image coordinates (u, v) with an undistorted lens is then given by

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} x' \cdot f_x + \frac{\gamma}{z} + c_x \\ y' \cdot f_y + c_y \end{pmatrix}. \tag{2.2}$$

This simplified model is not well-suited for most real-world camera lenses because it neglects lens distortions. Therefore, the nonlinear intrinsic parameters that describe the radial lens distortion with parameters k_1, k_2, k_3 and tangential lens distortion with parameters

p_1, p_2 are also included. With radius $r^2 = x'^2 + y'^2$, the correction vector τ for tangential distortion is

$$\tau = \begin{pmatrix} 2p_1x'y' + p_2(r^2 + 2x'^2) \\ 2p_2x'y' + p_1(r^2 + 2y'^2) \end{pmatrix}. \tag{2.3}$$

The distortion effects are corrected with:

$$\begin{pmatrix} x^* \\ y^* \end{pmatrix} = \begin{pmatrix} x' \\ y' \end{pmatrix} \cdot (1 + k_1 \cdot r^2 + k_2 \cdot r^4 + k_3 \cdot r^6) + \tau. \tag{2.4}$$

The corrected coordinates x^* and y^* can then be used as input in Equation 2.2 instead of x' and y'.

Extrinsic Parameters for World-To-Camera Transformations

The extrinsic parameters describe the rotation and translation of the camera coordinate system relative to the origin of the world coordinate system and are required to transform points between these two coordinate systems. Let R be the 3×3 rotation matrix and let T be the translation vector. The world coordinates of point $p^* = (x, y, z)$ are then transformed to camera coordinates of point p with

$$p = R \begin{pmatrix} x \\ y \\ z \end{pmatrix} + T. \tag{2.5}$$

Given point p in the camera coordinate system, it can then be projected into image coordinates as explained by Equations 2.2 and 2.4.

Calibration of Video Cameras

A camera is calibrated by solving the projection function for the unknown camera parameters. To solve these equations, correspondences between points in the world and their images are required. Methods for calibrating video cameras can be divided depending on how these point correspondences are acquired.

Methods with an explicit calibration object use an object of well-known size that usually shows a distinctive pattern from which reference points are extracted. For these points, either their absolute 3D coordinates or the relative positions between them must be known. Calibration methods with a calibration object can be further divided into planar and

noncoplanar methods. Planar methods require that all points lie on a plane. Examples of planar calibration methods are given by Sturm and Maybank [154] and Zhang [186, 187]. The technique proposed by Zhang [186, 187] is arguably the most often used method for camera calibration in computer vision research.

Noncoplanar methods require that not all reference points are on the same plane. Examples include Heikkila [76] where a 3D calibration object is used and Tsai [161] and Weng *et al.* [175] where a 2D calibration object is moved to exactly known locations. These methods are more complex than planar methods because they either require building a 3D object or measuring multiple reference positions relative to the camera. In contrast, for planar methods, it is sufficient to freely move a 2D object in front of the camera.

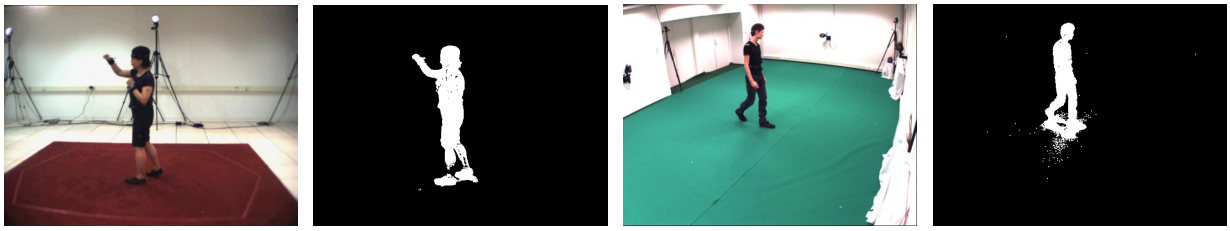
Methods without an explicit calibration object establish point correspondences between consecutive frames either by moving the camera, by following moving objects, or by analyzing the scene geometry. Examples include Faugeras *et al.* [62] and Cipolla *et al.* [54]. In general, methods that use well-known calibration objects result in more precise camera calibrations.

Given the intrinsic parameters that allow transformations between the local camera coordinate system and the image plane, the extrinsic parameters can be computed. The extrinsic parameters describe the position and rotation of the camera relative to a global world coordinate system. They are estimated by solving Equation 2.5 for the unknown rotation and translation parameters. The required point correspondences between image points and calibration object can be established in the same way as for the calibration of the intrinsic parameters.

Calibration of Depth Sensors

A depth sensor or depth camera gives for every image pixel the distance to the camera. Examples of depth sensors are stereo cameras, Time-of-Flight cameras, and sensors using structured light. Often, the intrinsic parameters are already given and the depth of each pixel can be transformed into the local 3D camera coordinate system. Therefore, this section focuses on the calibration of the extrinsic parameters.

Basically, the extrinsic calibration is done similarly to video cameras with known reference points in the world coordinate system. Depending on the depth sensor, these reference points are extracted with different methods. This work uses the Microsoft Kinect, a depth sensor based on structured light in infrared, as an example for depth cameras. Therefore, the methods described here focus solely on the Kinect. However, some of these methods



(a) Camera image (b) Foreground (c) Camera image (d) Foreground

Figure 2.1: Examples of silhouette images computed with foreground segmentation. The two left images show examples from the HumanEva-I dataset [148] with ViBe [33] and the two right images from the UMPM dataset [162] with thresholded background subtraction.

can also be used for Time-of-Flight cameras. For stereo cameras, the same methods as described for video cameras can be used.

When calibrating infrared sensors, like the Microsoft Kinect, reference points of the calibration object must also be visible in infrared or distinguishable in the depth image. Smisek *et al.* [152] illuminated a checkerboard pattern with a halogen lamp to make the pattern visible in infrared. Berger *et al.* [36] used a checkerboard with mirror surfaces for the black parts. These are not visible in the depth image, thereby generating the binary checkerboard pattern. Wilson and Benko used retro-reflectors with known 3D coordinates that are visible in the infrared image [177]. Tirpitz [18] manually selected a calibration object in the depth image and then estimated the extrinsic parameters. A method without a calibration object was introduced by Rösch [17] where correspondences between automatically extracted human joint positions were used to estimate the transformation matrices between multiple depth cameras.

2.1.2 Silhouettes and Foreground Segmentation

The silhouette of an object describes its projection into an image. In computer vision, the silhouette is usually visualized in white on a black background. It is computed with segmentation algorithms of which there is a large variety to choose from. In the context of this work, silhouettes are computed with foreground segmentation algorithms. Examples of foreground segmentations are shown in Figure 2.1.

A foreground segmentation is a labeling task where each image pixel is either labeled as foreground or background. There is a large body of literature and overviews can be found in [34, 35, 125]. In general, foreground segmentation algorithms can be characterized

by how they represent the background model and how the distance to the background model is computed. The background model is typically described on pixel level, *e.g.*, with mean values or probabilistic predictions [160], stored samples [33], one or multiple Gaussians [153, 189] or with non-parametric models [61]. Given the background model, the difference to the currently observed image can be computed. Then, to decide if a pixel should be labeled as foreground or background, the difference is compared to either a threshold, to stored samples, or used as input for a labeling function. In this work, the expressions silhouette image and foreground segmentation are interchangeable. In general, the specific foreground segmentation algorithm is not important as long as it computes a binary foreground mask.

2.1.3 Visual Hull

Shape-from-silhouette approaches compute a 3D reconstruction of an observed object and generally require two different inputs: fully calibrated cameras as described in Section 2.1.1 and silhouette images of the object of interest that are the result of foreground segmentation as described in Section 2.1.2.

Martin and Aggarwal introduced one of the earliest shape-from-silhouette examples for volumetric reconstruction in [103]. With the visual hull concept, Laurentini gave a formal description of the possibilities and, in particular, limitations of shape-from-silhouette methods [92]. Laurentini introduced and defined the visual hull as the maximal object that results in the same silhouette from all viewpoints [92]. Given a set of viewpoints, the visual hull can be computed by volume intersection. Each viewpoint defines a cone formed by rays originating from the viewpoint and passing through the silhouette boundaries. By intersecting the cones of multiple viewpoints, the visual hull is formed. Figure 2.2 shows an example for two viewpoints. By increasing the number of viewpoints, the accuracy of the volumetric reconstruction can be increased. With a limited number of viewpoints, not all possible setups result in a minimal volumetric reconstruction as intersections from one viewpoint might remove more parts and result in a tighter fit than from another one. In [144], Shanmukh and Pujari investigated heuristics for optimal viewing directions. However, even with an unlimited number of views, the main limitation of shape-from-silhouette is the inability to reconstruct concave surfaces as they are not represented by the silhouette [92].

There are two main ways [67, 91] to compute and represent visual hulls: either with a surface-based polyhedron representation or with a volumetric representation based on voxels.

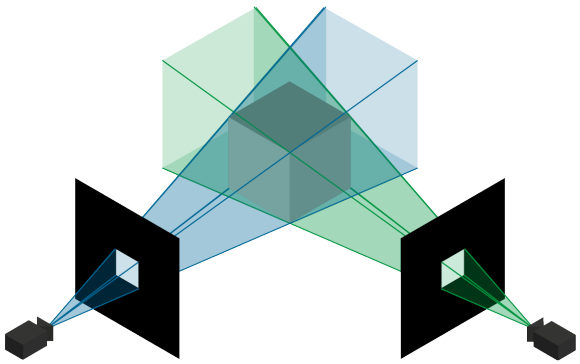


Figure 2.2: Volume intersection. By intersecting the camera cones that are defined by the silhouette boundaries of the observed object, an approximation of the 3D shape can be reconstructed.

Polyhedral Visual Hulls

A polyhedron is a 3D solid object with flat faces that are connected by straight edges. A polyhedral visual hull can be thought of as a solid block where parts were cut away by rays cast from each viewpoint through the silhouette contours.

Martin and Aggarwal [103] approximated the visual hull based on silhouette scan lines that pass through the silhouette contour. The intersection of scan lines from multiple views defines the volume segment which is later refined to create the final volume representation. In another work, Franco and Boyer [67] computed an exact polyhedral visual hull by connecting viewing edges of the silhouettes to the final mesh.

Visual hulls can also be computed in image space. In [104], Matusik *et al.* computed a polyhedral representation of the visual hull in image space for efficient computation. In [105], Matusik *et al.* computed the image-based visual hull directly to create a novel view without an explicit 3D representation.

Li *et al.* [96] showed that polyhedral visual hulls can be computed quite efficiently and they utilized hardware acceleration to achieve real-time runtimes.

Polyhedral visual hulls are well suited for visualization purposes [96, 105, 166], but depending on the silhouettes, their representation and computation can become quite complex. The continuous nature of this representation also makes it harder to use in certain algorithms due to the lack of primitive elements inside the volume. Therefore, for each 3D point, it must first be checked if it is inside the visual hull or not.

Volumetric Visual Hulls

In contrast to polyhedral visual hulls that are defined by their surface, volumetric visual hulls are defined by volumetric elements called voxels. The name voxel is composed of the two parts *volumetric* and *pixel*. As pixels are the atomic elements of images, voxels are the atomic elements of volumes. A voxel is usually a cube, but can also be represented with cuboids.

Volumetric visual hulls are computed by volume or space carving. The idea is to start with a solid volume for the area of interest and then removing all parts of that volume that do not belong to the visual hulls of foreground objects. This process is similar to the work of a sculptor removing parts of a granite block until the statue remains. When voxels are used as primitives, this process is called voxel carving.

Voxel carving approaches are attractive because they are computationally efficient. The basic voxel carving algorithm remains the same no matter of the specific algorithm being used. First, the volume of interest is subdivided into a discrete set of voxels. Then, every voxel is projected into each image. If a voxel does not fulfill a certain criterium, *e.g.*, project into the silhouette for all images, it is removed, or carved.

A lot of research focused on how the efficiency of voxel carving can be improved and several methods have been developed that reduce computations either by using special data structures, by precomputing voxel projections, or by accelerating computations with specialized hardware.

Voxel carving can be more efficient by avoiding unnecessary computations. In general, the silhouette occupies only a relatively small region of the image. This means that the majority of voxels are being removed and that these voxels are also close to each other. This can be utilized with an octree representation of the volume [106]. An octree is a tree-based data structure with the root node representing the whole volume. Each parent has eight child nodes with each child node having half the side length, or an eight of the volume, of their parent. The basic idea of octree-based voxel carving is that large volume blocks can be removed at an early step of the algorithm. If the voxel representing one parent node does not even partially overlap with the silhouette, it can be removed and all child nodes with it, thus avoiding a large number of computations. Examples of octree-based voxel carving can be found in Potmesil [131] and Szeliski [155].

The second approach trades computation time with storage space. One computationally expensive part of voxel carving is the projection of every voxel into each image. This computation can be mostly avoided by precomputed lookup tables that store these projections.

The main drawback of lookup tables is that their size grows cubic with the side length of voxels, *i.e.*, voxels with half the side length result in a lookup table with eight times as many entries. In addition, they are fixed to a predetermined resolution. Lookup tables can be used stand-alone or in combination with other approaches. Examples of lookup tables for voxel carving were shown by Luck *et al.* [100] and Kehl *et al.* [87].

The third approach is to reduce the computation time for voxel projections. Voxel carving is a highly parallelizable process because operations are exactly the same for each voxel. With access to affordable graphics hardware with highly parallelized processors, computations can be speeded up significantly. Four examples of GPU-based voxel carving algorithms can be found in Ladikos *et al.* [91].

The voxel carving approaches described so far assume perfect silhouettes without any defects. In reality, this is seldom the case. The next section describes methods to deal with silhouette defects.

Occlusions and Silhouette Defects

Imperfect silhouettes represent an object only partially and have a severe impact on the visual hull. Silhouette defects can occur in the presence of static occlusions where parts of the object are not visible. They can also result from errors in the foreground segmentation, *e.g.*, due to sensor noise, or because the foreground object is too similar to the background.

There are two strategies to deal with silhouette defects. Either the voxel carving method is relaxed or the silhouette image is modified.

Kim *et al.* [89] modeled the reliability of foreground and background pixels. If a voxel projects only into one background region and this region is marked as not reliable, the voxel is not carved. However, this still requires reliable foreground segmentations. Ladikos *et al.* used manually annotated, binary occlusion masks that were added to the foreground segmentations [91]. Guan *et al.* automatically built binary occlusion masks for static occluders [74]. Voxels projecting onto pixels of the occlusion mask do not result in the voxel being carved. But binary occlusion masks do not lead to minimal visual hulls, in particular because it cannot be decided if the voxel is in front or behind the occluder. In [73], Guan *et al.* improved the generation of occlusion masks by using a Bayesian sensor fusion approach that gives 3D information. However, this required long computation times of up to one minute per frame.

For real-time applications, a mechanism is required to accurately include 3D information of static occluders to compute minimal visual hulls without slowing down computation speed.

In this work, such a real-time voxel carving algorithm with 3D occlusion maps will be presented in Section 3.1. But even with accurate occlusion maps, foreground segmentation will seldom be perfect and approaches using voxels must be able to deal with imperfect data.

This concludes the volumetric visual hull computation with colorless voxels. The next section gives an overview of colored voxels before discussing depth carving.

Voxel Coloring

In the approaches described so far, voxels are colorless volumetric objects and only binary silhouette images were used. However, voxels can also represent color information. Seitz and Dyer [142, 143] directly reconstructed a 3D scene by coloring voxels and checking their consistency across multiple views and Culbertson *et al.* [56] introduced a generalized voxel coloring method. Voxels can also be colored after carving. Caillette and Howard [46, 47] decided for each voxel if it should be carved based on a comparison of a background model to pixels belonging to the voxel projection. If the voxel is not carved, it is assigned the color of these pixels. However, even though it might be beneficial to maintain color information in the volumetric reconstruction, voxel coloring has several drawbacks. It is only meaningful to color voxels on the surface of the visual hull, not inside the volume. This requires some form of visibility check for each voxel, *e.g.*, with a z-buffer. Also, colors of the same object can differ between camera views, *e.g.*, because of varying shutter settings. Further, depending on the size of voxels and the accuracy of silhouettes, the same voxel can project onto different regions which leads to washed-out colors for heavily textured objects.

Depth Carving

The visual hull is the maximal shape generating the observed silhouette [92]. With limited views, this implies that the visual hull is no tight fit to the actual shape. The reason is the lack of actual depth information for the binary silhouette images. Even though color carving can result in more accurate reconstructions [142], they encounter similar problems for non-textured regions.

The voxel carving variations described above project voxels into images that are typically recorded by video cameras. Besides video cameras, another type of sensor can also be used for voxel carving: depth sensors. Instead of color information, a depth sensor gives for each pixel the distance of the closest object at that pixel to the camera center.

There are various kinds of depth sensors that are based on different principles. Stereo cameras compute the disparity map of two images with the help of epipolar geometry. Time-of-Flight cameras measure the time it takes a light pulse emitted by the camera to return to it. Depth sensors based on structured light project a well-known point pattern and compute the depth based on the observed offsets. A popular example of this sensor type is the Microsoft Kinect.

The voxel carving algorithm developed in this work supports both camera as well as depth images. Section 3.1 compares the two approaches and shows how voxel carving can be improved with the use of multi-view depth images for tighter visual hulls.

2.2 Superpixel and Supervoxel Segmentation

Image segmentation is a fundamental task in computer vision and many algorithms require some kind of segmentation as a preprocessing step. The focus of the segmentation methods described here is on reduction of the input data. In the context of this work, this reduction will drastically improve the runtimes for human pose estimation. This section first introduces superpixel segmentation as a way to reduce the number of pixels in an image by grouping them. Then, supervoxel segmentation is explained as a continuation of superpixels in the domain of 3D data.

2.2.1 Superpixel Segmentation

The term superpixel segmentation was first described by Ren and Malik [181]. A superpixel segmentation partitions an image into a set of equally sized, non-overlapping and homogeneous regions. The segmentation of the image is guided by a similarity measure by which the pixels are grouped.

Superpixel segmentations, or short *superpixels*, have two characteristics in common independent of the specific algorithm. First, superpixel segmentations belong to the class of oversegmentation algorithms, *i.e.*, they partition an image into more segments than there are objects. Second, superpixels align well with what humans perceive as meaningful object boundaries. These two properties make superpixels particularly useful for applications that use them as building blocks or atomic primitives instead of pixels.

Superpixels are, in general, an intermediate representation and the main advantage is the reduction of the number of pixels for following processing steps. For example, an image

with VGA resolution with 640×480 pixels already has $307,200$ pixels. Grouping these pixels into only a couple of hundred superpixels reduces the number of inputs by three orders of magnitude.

Superpixel Algorithms

Superpixels group pixels based on a similarity measure. As there are many ways to measure the similarity and group pixels, there is a large variety of superpixels algorithms. Each algorithm has its own specific characteristics that determine its usefulness for certain application areas. Superpixel segmentations can be classified by how they initialize the segmentation, measure the similarity, and assign pixels to superpixels.

Ren and Malik [181] applied normalized cuts [101, 145], a graph-based segmentation technique, guided by features motivated by the Gestalt principles, to compute the superpixel segmentation. The normalized cuts approach was also used by Mori [114] for superpixel segmentation. The main drawback of normalized cuts is its high computational complexity which makes it inefficient, especially for larger images.

Felzenszwalb and Huttenlocher proposed another graph-based image segmentation algorithm in [63] based on minimum spanning trees. Even though their algorithm computes an oversegmentation of an image, it does not aim to compute superpixels. In particular, the shapes, sizes, and numbers of image segments varies greatly which does not match the definition of superpixels. However, their algorithm is often used for comparison with superpixel segmentations because of its high segmentation accuracy. Similarly, superpixels can also be computed with mean shift [55] and watersheds [164] even though they were not explicitly designed for this task.

Veksler *et al.* [163] formulated the superpixel segmentation in an energy minimization framework and solved it with graph cuts. Zhang *et al.* [185] used pseudo-boolean optimization as an improvement of [163]. TurboPixels, proposed by Levinshtein *et al.* [95], guides seeded superpixel growth with geometric flows. The segmentations are accurate, but due to the many parameters difficult to control. Inspired by TurboPixels, Xiang *et al.* [180] introduced a variation using eigen-images and Zeng *et al.* [183] proposed a structure-sensitive superpixel segmentation using the geodesic distance. Perbet and Maki introduced a superpixel segmentation based on random walks in [123] with an extended version by Maki *et al.* in [124]. Their algorithm includes a compactness measure to enforce spatial homogeneity. One drawback of these methods is that their runtime is in the range of seconds which is too slow to be useful as preprocessing for real-time applications.

The Entropy Rate Superpixels proposed by Liu *et al.* [97] achieve very good results on current metrics. They segment superpixels using an objective function that includes an entropy rate term for random walks and a balancing term. However, the resulting superpixels look quite irregular and lack smooth boundaries, even for relatively homogeneous image parts.

SLIC, simple linear iterative clustering, proposed by Achanta *et al.* [19] distributes superpixel seeds with respect to gradient information and clusters pixels based on k-means using color and Euclidean distance. It is very fast and achieves very good segmentation results which might be one reason why it inspired many other algorithms, *e.g.*, [121, 172] and [2]. Achanta *et al.* introduced variations of SLIC with an extensive comparison to other algorithms in [20].

In general, the segmentation into superpixels destroys the regular lattice structure of an image and neighborhood relations must be represented by graphs. This can have a negative impact on algorithms using superpixels because they cannot exploit the regular image structure, but must handle more general topologies. Moore *et al.* introduced Superpixel Lattices in [112] where the final superpixel segmentation is guaranteed to conform to a lattice structure. In [111], Moore *et al.* applied graph cuts to compute the superpixel lattice as an improvement to the greedy solution in [112].

Superpixel segmentations are not limited to color or grayscale images. Weikersdorfer *et al.* [172] applied a variation of SLIC [19] to RGB-D images, *i.e.*, registered color and depth images, recorded with a Microsoft Kinect. The main advantage is that their depth-adaptive superpixels have the same size in 3D independent of their position in the image.

In the context of this work, a superpixel segmentation algorithm similar to SLIC, but based on boundary evolution was developed. It offers a transparent way to control the compactness of the segmentation in [2]. The term compactness in the context of superpixel segmentations will be explained in the next section. In addition, the GPU implementation runs in real-time. The algorithm was extended in [4] to also guarantee that the superpixels conform to a regular lattice structure.

Superpixel Compactness

Compactness in the context of superpixels means that the superpixels have a regular shape with smooth boundaries. It is not only an aesthetic property, but also has implications on the accuracy and usefulness of superpixel segmentations. Many authors agree that compactness is a desirable property of superpixels and that superpixels should be compact [19, 95, 97, 111,

163, 183, 185, 123]. However, except Perbet and Maki [123, 124], superpixel compactness was not measured by other authors.

In [3, 4], the compactness of superpixels was discussed and a metric to measure the compactness of a given superpixel segmentation developed. These results will be presented in detail in Section 3.2.1.

Applications of Superpixels

Superpixels are designed to be useful as atomic primitives in applications. By working with superpixels instead of pixels, the number of inputs is significantly reduced, thus resulting in improved computational performance.

Given that superpixels originate from the domain of image segmentation, there are many segmentation applications that benefit from them. Lucchi *et al.* [99] applied superpixels to segment medical images and demonstrated a high accuracy labeling cellular structures. Achanta *et al.* [19] showed better object recognition results with their superpixel algorithm SLIC compared to other superpixel algorithms. Kohli *et al.* [90] applied superpixels to conditional random fields for image labeling tasks. Fulkerson *et al.* [69] segmented and localized objects with superpixels and superpixel neighborhoods. Levinshtein *et al.* [95] compressed and restored images with superpixels and showed compelling qualitative results. All authors argued that superpixels are beneficial for their image segmentation and recognition applications.

Superpixels have also been applied to analysis of video sequences. Ayvaci and Soatto segmented motion in videos by minimizing an energy function defined on the superpixel graph in [31]. Wang *et al.* demonstrated in [171] that their tracking framework based on superpixels is able to handle challenging scenarios with large changes in appearance and motion as well as occlusions.

Highly complex tasks like human pose estimation in images can in particular benefit from superpixels. Mori *et al.* [114] detected human half-limbs that are represented by superpixels to estimate the human pose. In [113], Mori used the superpixel graph to infer articulated poses. Both approaches are working with challenging sports images.

In the context of this work, superpixels were applied to automatic image segmentation by using them as building blocks in a segmentation-by-composition framework [7]. In another work [2], all benchmark foreground segmentation algorithms that were given in a competition could be improved for all metrics simultaneously by post-processing them with probabilistic superpixels in Markov random fields.

These examples across various domains of computer vision show the usefulness of superpixels to improve a wide range of different applications. The next section extends the concept of superpixels to the domain of 3D data with supervoxels.

2.2.2 Supervoxel Segmentation

Supervoxels are the continuation of superpixels in the domain of 3D data. Most of the ideas and methods for 2D images described in Section 2.2.1 can be directly applied to volumes as long as they conform to a regular 3D grid with dense data. There are two main ways of building these segmentation volumes: either by stacking images or by using 3D data.

The motivation of supervoxel segmentation is analogous to the motivation of superpixels. The segmentation decreases the number of primitives for following processing steps. In addition, supervoxels can describe the motion of object parts when applied to videos or their spatial expansion when applied to 3D data.

Image Stacks

Segmentation volumes can be constructed by stacking single images on top of each other. The images typically originate from a video sequence [163, 182], but there are also examples of medical scans at different depths [28, 98]. In both cases, the depth dimension is expressed in pixels. Typically, the distance between pixels of consecutive frames that are at the same position is one.

Veksler *et al.* [163] applied their superpixel algorithm to stacks of video frames and showed a high temporal coherency of the resulting supervoxels which is useful for tracking objects over time. In [19], Achanta *et al.* showed that SLIC can also be applied to supervoxels for video segmentation. Weikersdorfer *et al.* advanced the depth-adaptive superpixels from [172] to depth-adaptive supervoxels in [11]. Even though depth information is incorporated into supervoxels, they built the volumes by stacking video frames. An evaluation of several supervoxel algorithms for video processing was done by Xu and Corso [182]. They also adapted the superpixel measures boundary recall, undersegmentation error, segmentation accuracy, and the explained-variation metric to supervoxel segmentations of video sequences.

Supervoxel segmentations have also been applied to medical scans. Andres *et al.* [28] used watersheds in 3D to segment neural tissue scans into supervoxels. Lucci *et al.* [98] used supervoxels based on SLIC to segment mitochondria in stacks of medical scans.

3D Data

While the previous section described applications of supervoxels to artificial volumes, they can also be used directly on 3D data. There are segmentation techniques for 3D point clouds, for example Rabbani *et al.* [132] and Rusu *et al.* [139], but none of them compute supervoxels. During the time of this writing, there is only one supervoxel segmentation algorithm available for sparse 3D data.

Papon *et al.* [121] developed a supervoxel segmentation for 3D point clouds. The point clouds originate from RGB-D cameras like the Microsoft Kinect. Their algorithm works directly in 3D and images of multiple calibrated RGB-D camera views can be combined. The supervoxels are optimized for surface 3D data and the similarity is measured in a 39 dimensional space combining color, Euclidean distance, and normal information represented by Fast Point Feature Histograms [137]. The final similarity measure is given by a weighted sum of these three parts. In contrast to SLIC [19], supervoxels are computed flow-based, similar to a breadth-first search from the supervoxel center, to ensure connectivity. This, however, places restrictions on a potential GPU implementation.

Even though [121] segments 3D point clouds into supervoxels, that approach is optimized for 3D surface scans. Further it does currently not run in real-time which is required for human-computer interaction. Section 3.3 presents a real-time supervoxel segmentation algorithm for volumes that was developed in this work.

2.3 Pose Estimation and Articulated Body Tracking

This section gives an overview of human pose estimation and articulated body tracking. A pose describes the configuration of an object and the process to determine the parameters of a specific configuration is called pose estimation. In the context of this work, a pose always refers to a human pose unless specified otherwise. Pose estimation is a timeless process because it does not rely on temporal information. If temporal information is included, it is commonly referred to as tracking. Tracking human poses that are highly articulated is called articulated body tracking.

Both pose estimation and articulated body tracking are strongly linked to each other and therefore reviewed together in the following parts of this section. In Section 2.3.1, an overview of existing surveys and reviews shows possible classifications and identifies current challenges. A classification of pose estimation and body tracking approaches is given in Section 2.3.2. Section 2.3.3 discusses 3D pose estimation and tracking methods in more

detail before Section 2.3.4 gives an introduction to pictorial structures that are the basis for the 3D pose estimation and tracking system presented in this work.

2.3.1 Overview of Application Areas and Challenges

Pose estimation and articulated body tracking are active areas of research with many contributions. Therefore, an overview of existing reviews first shows possible classifications and taxonomies to group and structure different approaches. In addition, a summary of the main challenges shows the shortcomings of existing approaches. Based on these insights, pose estimation and body tracking methods that are related to this work are then examined in more detail.

Visual Analysis of Human Movement In his survey from 1999, Gavrila [71] divided methods to analyze full-body movements and hand gestures into 2D approaches with and without explicit shape models and into 3D approaches. Gavrila mentioned several applications that can benefit from human motion analysis, among them human-computer interaction, motion analysis, surveillance systems, but also virtual reality and video-compression through model-based coding.

Regarding outlook and future work, a number of challenges including the acquisition and usage of ground truth data for comparison, occlusion handling, and robustness are listed. Gavrila also highlighted the need for automatic initialization of model parameters, *e.g.*, limb lengths, as well as automatic initialization of the starting pose for tracking.

Computer Vision-Based Human Motion Capture Moeslund and Granum [109] presented a review with focus on human motion estimation in 2001 in which they discussed approaches from 1980 to 2000. They provided a taxonomy for human motion estimation based on four successive steps [109]: 1) initialization, 2) tracking, 3) pose estimation, and 4) recognition. Initialization is an important part for many approaches and is required for autonomous real-world applications, but also necessary for, *e.g.*, failure recovery. Due to the focus on motion estimation, in contrast to static pose estimation, tracking plays also an important role in their taxonomy.

Moeslund and Granum [109] identified three major application areas for motion estimation, namely surveillance, motion analysis, *e.g.*, for sports or medical purposes, and human-computer interaction. The performance of systems is compared in terms of robustness, accuracy, and speed. At the time of this review (2001), the field of motion estimation was

in an early stage of development [109] and challenges included, among others, a reduction of the number of assumptions and the need to use more training data. Given current systems that work in real-life scenarios and that use exhaustive amounts of training data [146], at least the last challenge can be seen as fulfilled. Hence, it could be an interesting direction to build systems that generalize well without the need of exhaustive training.

In [110], Moeslund *et al.* published a follow-up to their review in [109] that discusses approaches in the time between 2000 and 2006. The taxonomy remained the same as in [109] and consists of the categories initialization, tracking, pose estimation, and recognition. The same holds for the three application areas. Moeslund *et al.* [110] identified significant progress in the field of motion estimation and modified their proposed directions for future research. Besides a necessary increase in accuracy and robustness, they pointed towards the need for more general models for motion and behavior understanding. The large progress in performance was traced back to new part-based approaches for pose estimation, but the authors also stated that these must become more invariant, *e.g.*, for different viewpoints or appearances [110].

Vision-Based Human Motion and Action Recognition In 2007, Poppe [129] divided human motion estimation into model-based and model-free approaches. For model-based approaches, Poppe further distinguished between two phases: a modeling and an estimation phase. The modeling phase deals with the construction of a likelihood function that is then used in the estimation phase to find the most likely pose given this function. These top-level classifications are then further broken down into more specific and task-dependent categories.

Poppe sees many applications for human motion analysis, but particularly highlights surveillance, human-computer interaction, and automatic annotation (for example of videos) [129]. Similar to Moeslund and Granum [109], he identified machine learning and reducing the number of assumptions being made for motion analysis approaches as important future steps.

3D Human Pose Estimation and Activity Recognition In 2012, Holte *et al.* [78] presented a review of 3D human pose estimation and tracking approaches that use both a body model and multi-view video data with focus on recognizing different activities. They identified four general processing steps for pose estimation and used these in their taxonomy: 1) capturing vision data to 2) extract useful features, either in 2D or 3D, that are then 3) used for pose or motion capture to recognize either high level information, *e.g.*, a specific activity, or the full pose or motion with 4) an (optional) body model [78]. For

multi-view pose estimation, they identified common steps consisting of camera calibration and capturing, followed by voxel reconstruction that is used as input in a cycle of a) initialization and segmentation, b) modeling and estimation, and c) tracking whereas none of these three steps is mandatory and can vary from system to system.

Regarding 3D data acquisition, they identified the following shortcomings and challenges [78]: Volumetric reconstruction with shape-from-silhouette approaches, that are quite common for 3D pose estimation, is prone to errors due to imperfect silhouettes. Methods must be made more robust to deal with these kinds of defects. Further, occlusions and self-occlusions are a problem that can be partially solved with an increasing number of cameras; however, it is preferable if methods do not rely on too many cameras or high camera resolutions for good results. Finally, multiple cameras allow useful volumetric reconstructions, but their installation requires a lot of space. More compact setups would be preferable.

Further Reviews The reviews introduced above are a selection. Other reviews are now shortly summarized that are either similar to the reviews listed above or that focus on specific subproblems.

An early review from 1994 by Aggarwal *et al.* [24] discusses the motion of articulated and elastic non-rigid objects. They concluded that non-rigid motion estimation is still at a very early stage. They pointed towards gesture recognition applications, but did not restrict the motion estimation to humans and also mention applications like material deformation analysis. A following review from Aggarwal and Cai [23] from 1999 highlights applications that are more in accordance with other reviews from that time, namely surveillance, sports analysis, and human-computer interaction, but also content-based video labeling. The review is structured into three categories: model and non-model based body analysis, single and multi-view tracking, and recognition with template matching and state-space approaches. Challenges discussed include the presence of too many assumptions and the trade-off between runtime and accuracy.

Wang *et al.* [170] presented a review in 2003 with focus on detecting people, tracking, and understanding their activities. As major applications areas, surveillance, human-computer interaction, and motion analysis, *e.g.*, for sports or medical care, are mentioned. Identified challenges include occlusion handling and improving speed, accuracy, and robustness.

In a review from 2007, Werghi [176] focused specifically on estimating the human body shape from 3D scan data. In contrast to other reviews and due to the special focus on 3D scans, Werghi saw potential applications for clothing design and human factor engineering, but also virtual reality applications and human analysis, *e.g.*, for health care. In general,

3D scan data give a high-quality full view without occlusions that leads to a different focus of research. Werghi classified recent work based on landmark detection, scan segmentation, shape modeling, and tracking [176].

Ji and Liu [84] presented a review on view-invariant motion analysis in 2010 and categorized approaches by three criteria: detection of humans, view-invariant pose representation including estimation with and without a body model, and behavior understanding [84]. Specifically mentioned application areas are visual surveillance, motion analysis for sports, and video retrieval based on recognized actions.

Action and Activity Recognition The focus of this work is on pose estimation. In contrast, action and activity recognition deal with the problem of recognizing, classifying, labeling, and categorizing human motion. Most of the reviews mentioned above include a chapter about this topic because it is a direct application of motion estimation [71, 78, 84, 109, 110]. Other reviews with a stronger focus on action and activity recognition are given, for example, by Poppe [130] and Weinland *et al.* [174].

Summary This section presented an overview of reviews between 1999 and 2012 with a description of the developed and applied taxonomies, application areas, and key challenges. The key insights will now be summarized.

For pose estimation and body tracking, the most often mentioned application areas remained largely the same over the years and can be represented by three distinct groups. First, surveillance applications are very prominent because they benefit strongly from an automatic analysis of video streams. Here, the 2D pose estimation and tracking approaches are dominant as very often only one camera is available. The second group is motion analysis, either for sports or for medical purposes. Motion analysis is a direct application of body tracking with varying levels of difficulty ranging from coarse estimation of the human skeleton with 3D scan data to the complex estimation of athlete poses in single-view videos. Approaches in this group very often follow a model-based approach where a human body model is actively used. The third group of applications targets human-computer interaction, mainly with the purpose of actively controlling a system. Specific examples include entertainment applications and interfaces in smart environments. Here, approaches usually rely on 3D data and are required to run robustly in real-time.

Regarding taxonomies, the reviews showed that there is a wide range of different criteria to classify pose estimation and body tracking approaches. However, as Holte *et al.* showed in [78], all approaches must follow three to four specific processing steps that can be used

for classification. First, all approaches must acquire data with *sensors*. Second, from this data, *features* must be extracted. Third, these features are then used to *estimate or track* the pose. Fourth, in this process, a *body model* can optionally be used [78]. These four processing steps will be the basis for the overview in the next section.

The key challenges identified in the reviews vary over time. During the time of earlier reviews, the acquisition of ground truth data was more difficult and, consequently, there existed fewer datasets [71, 109]. Today, this problem seems to be largely solved as there are many datasets available for various purposes. Also, the lack of training data and machine learning algorithms was identified as a problem in earlier reviews [109, 129]. Today, with approaches that train detectors on approximately $500,000$ training images [146], this problem has been successfully addressed. More timeless challenges are real-time capabilities and robustness in general, as faster approaches with fewer errors are always preferable [71, 78, 109]. Occlusions and self-occlusions are seemingly timeless challenges even for more recent approaches [71, 78, 109, 129]. Further, increasing the invariance of approaches [110] to allow for fewer assumptions [71, 78] remains still a challenge for todays state-of-the-art systems.

2.3.2 Classifications of Pose Estimation and Body Tracking

This section gives an overview of specific approaches for pose estimation and tracking. It roughly follows the classification provided by Holte *et al.* [78] and categorizes approaches by four common steps: body model (I), sensor setup (II), features (III), pose estimation and body tracking algorithms (IV). These steps are then further subdivided and examples of approaches discussed, partly inspired by the reviews discussed above, in particular [71, 109, 110, 129]. Even though diverse approaches are discussed to give a broad overview, the focus is on 3D approaches. The overview starts with the body model because it is a very strong discriminating factor for the different approaches as it not only influences the estimation itself, but also the output of the system.

I Body Model

A body model describes the parts of the human body and their connections. Pose estimation approaches can be differentiated by whether or not they use a body model. If they use a body model, they can further be differentiated by how they represent it to describe a pose or motion. Also, the model itself must first be acquired and its parameters initialized.

Approaches without a Body Model Approaches without a body model assign a class label to the observed data. They are rather rare because they are limited by the number of classes they can recognize. Also, their accuracy depends on how similar observations are to stored samples. Howe [80] used lookup tables in the first step of his algorithm to retrieve the 3D pose for silhouettes and measured the similarity between observed silhouettes and stored examples. To overcome limitations of pure lookup-table based approaches, Howe additionally applied temporal smoothing to the predictions with Markov chaining.

Approaches with a Body Model The body model describes the "flesh and bones". The human body can be well described by a set of rigid limbs, *i.e.*, the bones, that are connected by flexible joints. Depending on the model, it can also include a description of the shape. The approaches differ in the number of limbs, joint positions and degrees of freedom, and if and how spatial extent is described. Most approaches use a body model and return its parameters as output.

The human body can be represented by a volumeless skeleton without any spatial description. Such a model only contains rigid limbs that are connected by joints and it can both be used in 2D as well as 3D. Advantages of this model are that it is simple, yet able to capture the most relevant information. It can also be represented as an acyclic graph which is beneficial for certain algorithms. Further, it can be used as an anchor in combination with spatial or volumetric representations as discussed below.

In 2D, the spatial dimension of the model can be described with 2D shapes, *e.g.*, rectangles or trapezoids. For example, Huang and Huang [81] described the limbs of the 2D body model with trapezoids that are linked together at the joint positions.

In 3D, the model parts can also describe the volumetric properties of the body. This is usually done with volumetric primitives, *e.g.*, ellipsoids, or surface representations, *e.g.*, polygon meshes. Mikić *et al.* [108] represented the torso with a cylinder and other body parts with ellipsoids to match voxel data. The parts are connected based on the twists body model [116]. In Cheung *et al.* [53], the body parts are represented by ellipsoids that are fitted to a surface voxel reconstruction. In another work, Cheung *et al.* [52] used rigid limbs that are connected by joints and additionally surrounded the limbs with colored surface points that originate from the surface of the visual hull. Kehl and Van Gool [88] used superellipsoids, a special form of superquadrics, to represent body parts. Plänkers and Fua [127] described the body model in a multi-layered approach. The first layer is a skeleton. The second layer attaches ellipsoidal metaballs to the skeleton to model muscles and tissue. The third and last layer, the skin or surface mesh, is computed with B-spline

patches. The advantage is that skeleton transformations directly affect the other layers which leads to more realistic animations of movements.

A spatial or volumetric representation can help to reduce the number of potential poses. For example, there is an infinite number of ways to fit a volumeless skeleton to volumetric data which can be reduced with volumetric limbs. However, this representation also comes at the cost of additional parameters that must be initialized or estimated and that lead to an increased computational complexity.

Number of Parts The number of limbs or body parts and, in particular, the degrees of freedom vary greatly between different approaches. Usually, the model contains primitives to represent the torso, head, arms, and legs. However, not all body parts are always included. For example, some approaches completely neglect the head [45]. Also, even though most approaches model the arms and legs as two rigid limbs that are connected by a separate joint, there are also approaches that model the arms and legs as one rigid limb [53]. The granularity of body pose estimation usually ends at the lower limbs and both hands and feet are not modeled in detail. The joints typically include the neck, shoulders, elbows, wrists, pelvis, hips, knees, and ankles. End points without any degrees of freedom are usually the head, wrists, and ankles. The total number of degrees of freedom for the whole model varies from approach to approach. For example, Deutscher *et al.* [59] used a model with 29 degrees of freedom. In [21, 22], Agarwal and Triggs used a model with 54 degrees of freedom.

It is important that the body model contains sufficient details for the task at hand. However, an increasing number of parts and degrees of freedom quickly leads to computationally intractable models.

Model Acquisition and Initialization The human body model can either be given and fixed or build during pose estimation. Here, model acquisition and initialization refers to the kinematic structure of the human pose or the geometric objects representing body parts. This is different from the initialization of the starting pose that will be discussed later in the context of pose estimation algorithms.

In general, the body model is assumed to be given and fixed. Some approaches allow a certain flexibility for the sizes of body parts. For example, Mikić *et al.* [108] refined the sizes of the body parts in their model until they matched the observed voxel reconstruction. Mori [113] used occlusion variables to reason about occluded limbs. This helps to avoid unnatural poses that result from fitting body parts into the observation even though they

are not visible. It is also possible to use anthropometric ratios [60] to initialize the size of body parts. With these ratios it is sufficient to estimate the overall size of a person to infer the specific limb sizes.

There are also approaches that do not require the model to be known in advance but build it during estimation. For example, Felzenszwalb and Huttenlocher [64] showed that their pictorial structures approach can also be used to learn the model from training data.

II Sensor Setup

The sensor setup describes how input data is acquired. Approaches can be differentiated based on the sensor type or modality, the number of sensors, and their mobility.

Sensor Type There are various measuring principles that can be applied in sensors to acquire image data. Common sensor types are video cameras, depth sensors, and thermal cameras.

Probably the most common sensor type for pose estimation are video cameras that either give grayscale or color images. With special filters and sufficient light in the infrared spectrum, video cameras can also be used in the infrared spectrum. Examples for pose estimation with video cameras are given by Wren *et al.* [179], Deutscher *et al.* [59], and Felzenszwalb and Huttenlocher [64], to name just a few.

Depth sensors do not give a colored image of the observed scene, but a 2.5D depth scan. Here, for each pixel the depth information to the closest object is stored. Various modalities can be used for depth sensors, among them stereo cameras, Time-of-Flight cameras, and structured light sensors.

Stereo cameras compute a disparity image based on the offsets in two rectified images of two calibrated cameras. Plänkers and Fua [127] estimated the volumetric pose in 3D and used a stereo setup to compute 3D features. By using stereo, it is not required to distribute the cameras across the whole room, but a more compact setup can be used. Nickel and Stiefelhagen [119] and Nickel *et al.* [118] used stereo information to track the 3D position of the hand of a person for human-machine interaction. Azad *et al.* [32] proposed a system that tracks the upper body pose for human-robot interaction. Their approach uses both image and depth features extracted from stereo cameras in a particle filter framework.

Time-of-Flight cameras emit light and measure the time it takes the reflected rays to return to the sensor. Schwarz *et al.* [141] computed landmark features, like head and hands, with

the geodesic distance in depth images recorded with a Time-of-Flight camera. Ganapathi *et al.* [70] used data from a single Time-of-Flight camera. In their approach, information about body part locations is computed with part detectors and propagated upwards along the kinematic body model to capture the 3D pose.

Structured light sensors, *e.g.*, the Microsoft Kinect, project a well-known pattern of points and measure their offsets. Shotton *et al.* [146] developed an algorithm for the Microsoft Kinect that labels pixels based on depth features that are used as input for randomized decision forests.

Thermal cameras measure the heat emitted by objects and are not as often used for pose estimation as other sensor types. One example approach was presented by Iwasawa *et al.* [83] where a single thermal camera is used to extract the silhouettes of humans by thresholding the image.

The choice of the sensor type depends on the environment where the pose estimation approaches should be applied. Structured light sensors are relatively cheap and give good depth maps, but have problems in direct sunlight. Similarly, active sensors like Time-of-Flight cameras can interfere with each other which can be problematic for multi-camera setups. Video cameras have no problems with multi-view setups, but are sensitive to varying lighting conditions and shadows. Therefore, it is preferable if pose estimation approaches can work with different sensor types without modifications and if they are not limited to one single sensor modality. This would introduce flexibility and allow for various areas of application.

Sensor Number The number of sensors is also an important criterion for discriminating pose estimation and body tracking approaches. There are monocular or single-view approaches that require only one sensor and there are multi-view approaches with two or more cameras.

Monocular approaches with only one camera are common for 2D pose estimation. Examples for 2D pose estimation were presented by Mori *et al.* [114] and Mori [113] based on superpixels or the pictorial structures approach from Felzenszwalb and Huttenlocher [64]. However, it is also possible to reconstruct a 3D pose based on 2D images. Agarwal and Triggs [21, 22] estimate the 3D pose directly from monocular single images or image sequences with nonlinear regression based on features extracted from the silhouettes. Taylor [157] showed how 3D poses can be estimated from a single image by considering the foreshortening of body parts. Brauer and Arens [41] presented a modification of [157] that uses a more realistic camera model and also computes unique solutions. However,

monocular approaches often suffer from disambiguities, unknown scale, and self-occlusions that can be resolved with multi-view setups.

Multi-view approaches, for example with voxel carving, typically require calibrated cameras. Caillette and Howard [47] used between two and four cameras for voxel reconstruction, Cheung *et al.* used five cameras in [53] and eight cameras in [52]. Even though calibrated cameras might be the rule, they are not always a necessity. Rosales *et al.* [135] presented an approach with multiple uncalibrated cameras for 3D pose estimation. Hasler *et al.* [75] even used multiple moving and unsynchronized cameras to track a 3D mesh. The advantage of calibrated cameras is that they allow an accurate inference of 3D information. However, this comes at the cost of reduced mobility and increased sensitivity to disturbances.

Sensor Mobility The sensors can be either fixed to a stationary position or move around during the estimation process.

Stationary sensors are often required if assumptions are being made about the background or if the specific position of the camera is important. The latter is often the case for multi-view approaches where the calibrated cameras cannot be moved. Assumptions about a stationary background are typical if silhouette information is computed with foreground segmentation. For example, Brand [40] computed the silhouettes with a background subtraction algorithm that requires a static background. In particular, approaches that use voxel carving require that the cameras remain fixed during pose estimation as they are calibrated [47, 53].

Pose estimation with mobile sensors is typical for analysis of video sequences. For example, Ferrari *et al.* [65] estimated the body pose in sequences of the Buffy TV series. Dealing with mobile cameras is mostly an issue of initialization. Approaches that do not need temporal information and that can initialize themselves automatically can treat pose estimation with moving cameras as a frame-by-frame estimation problem. However, there are also multi-view pose estimation approaches that use multiple uncalibrated and moving cameras, as for example in Hasler *et al.* [75].

III Features

The features have a great impact on the properties and performance of pose estimation approaches. The dimensionality of feature vectors can vary, but the origin of the features can usually be differentiated into 2D and 3D data.

2D Features Features that are directly extracted from an image are referred to as 2D features. Examples of 2D features are color, edges, contours, and motion [129].

Color features can be described by different color spaces, *e.g.*, RGB or Lab, as intensities, or as texture. In their system Pfinder, Wren *et al.* [178, 179] used colored blobs to represent and track body parts.

Edge features represent large intensity differences between neighboring pixels. Dalal and Triggs [57] introduced the HOG descriptor, histograms of oriented gradients, and applied it to human detection. Ferrari *et al.* [65] used the HOG descriptor to detect upper bodies in videos to initialize their estimation algorithm. Wachter and Nagel [168] used edge information in the update step of an iterated extended Kalman filter to track a 3D person model.

Contour and silhouette features are usually computed with foreground segmentation. In [80], Howe used silhouette images to look up reference poses. Iwasawa *et al.* [83] computed the distance transform of a silhouette image to get the center of gravity and find candidates for extremities like hands and feet. Brand [40] used hidden Markov models to learn a direct mapping from 2D silhouette sequences to 3D models.

Motion features represent the movement of an object and can be measured by establishing correspondences between features in different images or by computing the difference of consecutive frames, similar to foreground segmentation. For example, Sminchisescu and Triggs [151] incorporated optical flow and motion boundaries into their cost metric to estimate human motion in 3D from monocular sequences.

Single image features can also be combined to more complex descriptors. This is advisable if features are complementary to each other. However, each additional feature also comes with increased computational cost and can potentially introduce errors that must be dealt with, *e.g.*, due to noise in the data.

3D Features Features extracted directly from 3D data or 3D reconstructions are called 3D features. They can be used in combination with 2D features, in particular color. Here, however, the focus is on features that are specific to the domain of 3D data.

The depth values of pixels can be used as features if depth sensors are used. Ziegler *et al.* [188] used a 3D point cloud generated with one or more stereo cameras to track the upper body. Shotton *et al.* [146] developed depth features based on the differences between pixels at various offset positions relative to the classified pixel. These features are then used in randomized decision forests to estimate human poses when viewed with a Microsoft

Kinect. Schwarz *et al.* [141] used the depth information to compute the geodesic distance on the surface of a human body relative to its center to compute landmark features.

Voxels are often used for multi-view 3D approaches. Cheung *et al.* [53] used voxel carving to reconstruct a person in 3D and fit an ellipsoidal body model to the observation. Caillette and Howard [46, 47] combined color and voxel information to track an articulated body model represented by colored blobs.

Features in 3D can also include motion information. Weinland *et al.* [173] introduced motion history volumes. They capture the change over time for voxel-based 3D reconstructions and allow view-point invariant action recognition.

IV Pose Estimation and Body Tracking Algorithms

Pose estimation gives the parameters of the body model or the pose class for the current observation or time step. Articulated body tracking additionally includes temporal information and measures the poses over time. Both approaches can also be combined. For example, pose estimation can be used for tracking initialization and failure recovery. Therefore, they are discussed together in this section.

The different approaches can be classified by how much a priori knowledge they use, whether they follow a bottom-up or top-down procedure, and, in particular for tracking approaches, whether they use single or multiple hypotheses. In case a body model is used, another differentiating factor is how it is initialized.

A Priori Knowledge Moeslund and Granum [109, 110] distinguished between model-free approaches and approaches with indirect or direct model use. In their work, the term model refers to the amount of a priori knowledge that is being used with respect to the body model. This means that model-free approaches can still output a body model.

Approaches with direct model use apply the model during the estimation process. Deutscher *et al.* [59] proposed an annealed particle filter and used it for pose estimation. The particles, *i.e.*, the body model hypotheses, are directly evaluated in the image. Carranza *et al.* [50] estimated parameters of the body model by minimizing an energy function that measures the overlap of the projected model with the observed silhouettes.

Approaches with indirect model use rather apply the body model as reference, *e.g.*, by using anthropometric knowledge about limb ratios [60]. Mikić *et al.* [107, 108], for example, used templates with expected body part sizes to fit volumetric primitives to voxel data.

Approaches without a priori knowledge extract features directly from the data without using a body model for guidance. Howe [80] directly computes the pose candidates with lookup-tables based on a silhouette. Part-based detectors can also be used without a priori knowledge. Bourdev and Malik [39] and Bourdev *et al.* [38] developed poselets to detect people. A poselet is a part-based descriptor that also includes information about the pose. Poselets can be used to detect body parts in an image without any a priori knowledge about the pose.

Using a priori knowledge about the body model to extract features can improve body tracking. However, then the question arises how the model is initialized and, in particular, reinitialized if tracking fails.

Top-Down and Bottom-Up Approaches can also be classified by how they begin the estimation process. They can either start from a top-down perspective or use bottom-up information to estimate the body pose.

Top-down approaches take the body model and measure how well it fits to the observation. They often use a given pose, *e.g.*, the track from a previous time step, and adjust it to the new observation. For example, Delamarre and Faugeras [58] used physical forces to push the model until it matched the observed silhouette. After initialization, the Pfinder system [178, 179] uses the predicted model to update blob statistics given the current image.

Bottom-up approaches, on the other hand, start with information extracted from the image and use it to estimate pose parameters. The implicit shape model proposed by Leibe *et al.* [93, 94] can be used to learn object parts and detect them in the image. Müller and Arens [115] showed how these implicit shape models can then be used to estimate the human pose. In following work, Brauer *et al.* [42] showed how the 3D pose can be retrieved by comparing the detected 2D landmark features with projected 3D joint positions of a stick figure model. Mori *et al.* [114] used segmentation into superpixels as a preprocessing step and then used torso and half-limb detectors to find body part candidates. In [64], Felzenszwalb and Huttenlocher presented their pictorial structures approach for object recognition. An object is represented by single parts that are then assembled with the pictorial structures approach. Another part-based approach was presented by Sigal *et al.* [149] where parts are assembled through inference over a graphical model.

Both strategies have their advantages and disadvantages. Top-down approaches generally require that an initial pose is available. Then, the search space can be drastically reduced. Bottom-up approaches often do not need any information about previous poses. However,

if the part detectors yield too many hits at ambiguous locations, it is difficult to obtain the correct solution.

Single and Multiple Hypotheses This classification is mostly relevant for body tracking approaches. It differentiates algorithms based on the number of hypotheses that are used.

Single-hypothesis trackers only maintain one hypothesis. This can be more efficient than keeping track of multiple hypotheses, but then failure recovery is generally harder. Usually, a re-initialization is required if tracking fails. For single hypothesis tracking, the Kalman filter can be used as in Kakadiaris and Metaxas [85].

Multi-hypotheses trackers maintain many tracks simultaneously. The key idea is that if one track gets lost or stuck in a local minimum, the other tracks can still be used to infer the correct estimation. A famous multi-hypotheses tracker is the particle filter [30] proposed by Gordon *et al.* [72], and in particular the condensation algorithm by Isard and Blake [82]. Deutscher *et al.* [59] proposed an annealed particle filter and showed that it is well suited to cope with the high number of dimensions involved in human pose estimation.

Initialization Here, initialization refers to the free model parameters, *e.g.*, joint angles and body part positions, and not the model structure itself, which has already been discussed above. Initialization is particularly relevant for tracking-based approaches and plays an important role for real-world systems. It determines their degree of autonomy and how they can recover from tracking failures. An initialization can either be manual, semi-automatic, or automatic.

Manual initialization requires input by the experimenters. These approaches are the least autonomous and cannot recover by themselves from tracking failures. Cheng and Trivedi [51] required manual initialization at the beginning of evaluation. Perales and Torres [122] offered an interactive mode where users can match the model with the observation for all frames. Manual initialization is mainly necessary for systems that focus on tracking or that require a very precise initialization. Most current approaches usually offer at least a semi-automatic initialization method.

Semi-automatic approaches require some form of interaction, but are otherwise automatic. For example, tracking can require the use of a key pose to initialize the first pose. Caillette and Howard [47] required that the user assumes a starfish position for initialization. Bernier *et al.* [37] initialized their tracker by detecting the face. They further assumed that the starting pose is facing the camera with the arms along the sides.

Automatic initialization is the most flexible solution and is required for autonomous real-world applications. Often, these are pose estimation approaches that also incorporate temporal information to improve the frame-based estimations. As approaches like the one proposed by Shotton *et al.* [146] and Taylor *et al.* [158] showed, systems with automatic initialization can successfully operate under real-world conditions.

Summary

This concludes the overview of pose estimation and body tracking approaches. Based on previous reviews summarized at the beginning of Section 2.3.1, in particular [71, 78, 109, 110, 129], selected approaches were discussed following four characteristics [78]: body model, sensor setup, features, and estimation and tracking algorithms. Even though there is a large variety of different approaches, these categories are well suited to capture their main characteristics.

Even though there are approaches that recover 3D poses from single images, multi-view setups are more typical for 3D pose estimation. This is in particular the case if a high computational efficiency is required [46, 53]. In case of multi-view approaches with calibrated and stationary cameras, many approaches compute an intermediate 3D reconstruction with voxels as a preprocessing step [46, 51, 52, 108], as was also done in this work. Here, however, pose estimation is based on supervoxel segmentations that reduce the number of voxels to further increase computational efficiency. Both multi-view 3D pose estimation algorithms and voxel-based approaches will be discussed in more detail in the following section.

In recent years, it was shown that real-time 3D pose estimation is possible with part detectors that were trained on exhaustive amounts of training data [146, 158]. Part detectors also play an important part in current 2D pose estimation approaches [38, 39, 93, 115]. However, the main drawback of such detectors is that they are generally biased through the training data. In contrast, this work will require little prior information by using segmentation as basis for pose estimation.

The pictorial structures approach has proven to be very powerful [45, 64, 65], in particular for 2D pose estimation approaches and in combination with part detectors. As this work will show an efficient solution for pictorial structures in 3D, it will be further discussed in Section 2.3.4.

2.3.3 Multi-View 3D Pose Estimation and Tracking

This section focuses on pose estimation and articulated body tracking in 3D, in particular on multi-view approaches. The approaches are classified by whether they combine information from multiple 2D images to compute a 3D pose or if they estimate the 3D pose directly on 3D data.

Combining Multiple 2D Pose Estimations

The approaches discussed in this section use information and features gathered from multiple views to infer the 3D pose. They either match a 3D body model to observed image evidence or combine multiple 2D pose estimates in 3D.

Hofmann and Gavrila [77] combined information from multiple 2D images to infer 3D poses in challenging scenarios. For each single image, first the silhouettes of persons are computed and then used to retrieve 3D pose candidates from a lookup table. The candidates of all views are then projected back into the single views and evaluated. The poses that best match the observations and that are temporally consistent with previous estimates, are then selected as output. While this approach achieves good results, it is relatively slow and takes approximately 30 to 40 seconds per frame [77].

Another approach that also uses image features was presented by Amin *et al.* [26]. They directly estimated the 2D poses in multiple views using a pictorial structures approach. Based on these multiple 2D estimates, they then recovered the 3D pose by triangulation. The drawbacks are that the detector performance relies on good training data and that they must be executed for each additional camera image.

Hasler *et al.* [75] estimated the 3D pose by aligning a 3D mesh with the observed 2D data from multiple camera views. They used multiple unsynchronized cameras and calibrated them based on structure-from-motion. This also allowed them to reconstruct the static background. Then, they minimized the difference between the observed contours and the contours of the projected surface mesh. Due to the high flexibility, the computational complexity is rather large. Further, the algorithm requires a mesh of the person as input, *e.g.*, by scanning them in advance.

Approaches that rely on features or estimations that are directly based on 2D images are generally more flexible in terms of the number, placement, and mobility of cameras. However, the feature extraction or pose evaluation is usually computationally expensive and, therefore, every additional view leads to an increase of computation time. Compared

to this, 3D reconstruction is computationally less expensive. Therefore, it often is beneficial for real-time approaches to directly work on 3D data.

Pose Estimation with 3D Data

Multi-view pose estimation can also be done directly on 3D data. These approaches generally use multiple calibrated cameras to reconstruct the observed scene in 3D, often with voxel carving (Section 2.1.3). The poses are then either estimated by growing volumetric body parts, by tracking volumetric primitives, *i.e.*, body parts, or by extracting volumetric features.

Growing Volumetric Primitives Given the 3D voxel reconstruction, the positions of body parts can be estimated by "growing" volumetric primitives based on the data. For example, Mikić *et al.* [107, 108] represented body parts as cylinders and ellipsoids and estimated their sizes by fitting them to a voxel reconstruction. They started this procedure by first detecting the head. This is done by fitting a spherical head template within expected head sizes to the voxels. Then, the torso template is connected to the neck and fitted to the voxels beneath it. A similar procedure is done with the four limbs for the largest remaining clusters.

An advantage of fitting approaches is that they often do not require an explicit initialization step. However, there are several drawbacks. For example, it is problematic to deal with body parts that are too close to the body. Such parts are likely to be incorporated into other and larger parts, *e.g.*, in case of arms and torsos. Further, a fitting approach relies on certain assumptions about the seed points. Here, for example, the whole approach relies on finding the head based on a spherical head template.

Tracking Volumetric Primitives The approaches described here initialize a body model with volumetric body parts to match the observed data and then track them over time.

Cheung *et al.* [52] fit ellipsoids, that represent body parts, to colored surface voxels. The body parts are then tracked over time. This system requires an initialization step where people are asked to move specific body parts to initialize the surface points of the model. A separate initialization phase is a common drawback if the tracking system relies too strongly on appearance information. Further, color-based appearance models are likely to change in case of varying lighting.

Caillette and Howard [46, 47] fit a body model to colored voxels and then track it over time. Their approach attaches colored blobs to a skeleton and tracks them using an expectation-maximization algorithm based on voxel data. Similar to [52], the model is initialized in a semi-automatic way, here by assuming a starfish pose. This is problematic in case the track gets lost because no automatic initialization is provided. Also, color-based approaches are prone to failure due to appearance changes.

Cheng and Trivedi [51] proposed to model volumetric body parts with Gaussian mixture models that are connected by a kinematic model. Their approach also uses voxel data and gives good results, but requires a precise manual initialization of the body model. This is a severe drawback for a real-world system where manual initialization is not an option. Further, the system does not run in real-time as a result of the complex model [78].

Canton-Ferrer *et al.* [48] proposed a tracking approach with an annealed particle filter. Body parts are modeled with truncated cones. A pose, *i.e.*, a particle, is evaluated using volumetric as well as surface features. Their body model has 27 degrees of freedom. This requires a large number of simultaneously tracked hypotheses, in their case $3,600$ particles, which is computationally expensive.

A common drawback of approaches that track volumetric body parts is the initialization phase. Most approaches follow either a manual or a semi-automatic procedure. For most real-world systems, however, it is crucial that they operate autonomously without requiring user cooperation or manual input.

Extracting 3D Descriptors It is also possible to extract information directly from voxels to get information about the current pose. For example, Sagawa *et al.* [140] proposed a system that uses a feature vector computed in the cylindrical region around the voxel reconstruction to look up pose candidates in a database. While this approach is fast, it is limited by the exemplar poses stored in the database. Further, assumptions are required about the orientation of persons as the axis of the cylinder passes upwards through the center of mass.

Summary Voxels are very popular for multi-view 3D pose estimation with stationary cameras and are used by all approaches discussed above. With voxels, the body model usually consists of volumetric primitives that are either automatically fitted to the data [107, 108] or initialized based on the data [46, 47, 48, 52]. As discussed, both approaches have drawbacks as they either rely on assumptions or require some form of user input. Both are problematic for real-world applications.

Most of the discussed approaches follow a tracking scheme using common algorithms like expectation-maximization [47, 51], Kalman filters [107, 108], or particle filters [48]. Regarding the tracking framework, it is important that the hypotheses can be evaluated quickly. This is not the case if the primitives are too complex or if too many hypotheses must be evaluated simultaneously.

Outlook The pose estimation algorithm presented in this work is designed for volumetric data (Section 2.1), but does not require volumetric body part representations. It uses a skeleton model in combination with supervoxels as primitives (Section 2.2) which implicitly encodes volumetric information. By using supervoxels, the search space is effectively reduced to a tractable dimension which allows for real-time single frame pose estimation. Also, the search space reduction allows to directly estimate the best pose without any specific initialization step.

As an additional benefit, the presented system does not rely on trained detectors or stored examples. Nevertheless, it provides a way to directly integrate additional information if it is available. This process can also be used to integrate temporal information for articulated body tracking in a seamless way, as will be discussed in Section 4. The algorithm is based on pictorial structures that will be explained in the next section.

2.3.4 Pictorial Structures

This section introduces the pictorial structures framework for pose estimation. It is the basis for many 2D pose estimation approaches. In this work, it will be extended to 3D with a focus on supervoxels.

A pictorial structure is a simplified representation of an arbitrary object. This representation consists of two elements: object parts and connections between these parts. For example, a pictorial representation of the face [64] could consist of five single elements representing the two eyes, two mouth corners, and the nose together with four connections between the nose and each of the other parts. The pictorial representation of an object is by no means unique and can be adjusted as necessary.

A pictorial representation is helpful to reduce an object to its core elements. Based on such a pictorial structures representation, Fischler and Elschlager [66] proposed a framework to match and find objects in images. They represented objects as single parts that are connected with springs. The optimal solution is given by the configuration that minimizes an energy function. The energy function combines an appearance term for each single part

and a binary term that describes the cost of modeling the connections between two parts. Their approach is very general which leads to a high computational complexity.

An efficient framework for pose estimation with pictorial structures was proposed by Felzenszwalb and Huttenlocher [64] and the overview given below follows their explanation in [64]. Their framework is based on [66], but requires that the pictorial structures are described by a tree graph, *i.e.*, the graph cannot have any cycles. Then, the energy minimization can be efficiently solved with dynamic programming. They also proposed solutions to automatically build the model using training data and to compute multiple good hypotheses instead of just one optimal one. For applications using pose estimation, it is desirable to have a specific human body representation and not an automatically generated one. Therefore, the model used in this work is not automatically estimated, but given. However, the proposed approach also allows to estimate multiple hypotheses.

The next two sections first describe the mathematical formulation of pose estimation with pictorial structures before giving example applications where pictorial structures have been used for pose estimation.

Mathematical Formulation

This section introduces the mathematical basis to formulate the task of pose estimation in the pictorial structures framework. It closely follows the work of Felzenszwalb and Huttenlocher [64] and largely adopts their notations.

Pictorial structures represent objects by atomic parts and connections between them. Let the object be represented by graph $G = (V, E)$ with vertices V representing parts and edges $e_{ij} \in E$ representing connections between two parts. A specific pose is then represented by the part configuration $L = (l_1, l_2, \cdots, l_n)$ where each l_i represents the location of a part $v_i \in V$. The body model itself is given by model parameters θ that describe, for example, the part appearances or the connections between parts.

The problem of finding the specific configuration that best describes the data can then be formulated in a statistical framework. The posterior distribution $p(L|\mathcal{I}, \theta)$ describes how likely it is to observe configuration L given image evidence $\mathcal{I}$ and the model parameters θ. With Bayes' rule, it can be written as

$$p(L|\mathcal{I}, \theta) \propto p(\mathcal{I}|L, \theta) \cdot p(L|\theta). \tag{2.6}$$

The distribution $p(\mathcal{I}|L,\theta)$ describes how likely it is to observe image evidence given a configuration and the model parameters, *i.e.*, it describes how well the appearance of the configuration fits to the image. The prior distribution $p(L|\theta)$ describes the likelihood of a specific configuration given the model parameters, *i.e.*, it describes how well the connections between parts fit to the model.

The problem of finding the optimal configuration L^* is then given by

$$L^* = \underset{L}{\operatorname{argmax}}\Big(p(\mathcal{I}|L,\theta) \cdot p(L|\theta)\Big). \tag{2.7}$$

With the simplifying assumption that the image evidence for a part is independent from the image evidences of all other parts, which is true as long as no occlusions occur, the distribution describing the image formation can be written as

$$p(\mathcal{I}|L,\theta) \propto \prod_{i=1}^{n} p(\mathcal{I}|l_i,\theta). \tag{2.8}$$

By using the fact that the model is described by a tree structure and by focusing only on relative positions between parts, the prior can be written as

$$p(L|\theta) = \prod_{(i,j)\in E} p(l_i,l_j|\theta). \tag{2.9}$$

Inserting Equations 2.8 and 2.9 into Equation 2.6 then leads to the final equation for the posterior distribution:

$$p(L|\mathcal{I},\theta) \propto \prod_{i=1}^{n} p(\mathcal{I}|l_i,\theta) \prod_{(l_i,l_j)\in E} p(l_i,l_j|\theta). \tag{2.10}$$

The formulation of Equation 2.10 is a typical starting point for energy minimization approaches. The energy function is given by taking the negative logarithm of Equation 2.10. The optimal configuration L^* is then the configuration that minimizes this energy. Let $m_i(l_i) = -log\ p(\mathcal{I}|l_i,\theta)$ describe the energy for each single part, *i.e.*, its appearance, and let $d_{ij}(l_i,l_j) = -log\ p(l_i,l_j|\theta)$ describe the energy of the connection between two parts. Then, the optimal configuration L^* is given by

$$L^* = \underset{L}{\operatorname{argmin}}\Big(\sum_{i}^{n} m_i(l_i) + \sum_{(l_i,l_j)\in E} d_{ij}(l_i,l_j)\Big). \tag{2.11}$$

Equation 2.11 represents the maximum a posteriori (MAP) estimate and can be efficiently solved with dynamic programming if the possible locations of parts can be restricted for the binary term. This is the second requirement for an efficient solution, in addition to the tree-structure of the object model [64].

In general, the computation of the prior, *i.e.*, term d_{ij}, poses the most problems from a computational perspective for approaches using pictorial structures. Let n be the number of parts in the model and h be the number of candidates for these parts in the image. The complexity to compute appearance term m_i is then linear in the number of candidates, *i.e.*, its computational complexity is in $\mathcal{O}(nh)$. However, the binary term d_{ij} requires, in principle, that the connections between all pairs of candidates are evaluated, *i.e.*, its complexity is in $\mathcal{O}(nh^2)$. Even though Felzenszwalb and Huttenlocher [64] reduced this complexity by limiting the possible numbers of locations, it still is very time consuming. In this work, it will be shown how special segmentation techniques can be used to further reduce the overall complexity.

The next section discusses current approaches that use pictorial structures in the context of human pose estimation and focuses on how the energy terms are evaluated and the complexity reduced.

Example Approaches

In addition to the efficient solution for the pictorial structures framework itself, Felzenszwalb and Huttenlocher [64] also presented two applications: face detection with Gaussian derivative filters to model the appearance of facial landmark features and human pose estimation with silhouette images. Both approaches are used for 2D images and demonstrated how the pictorial structures framework can be used in an efficient way for two quite different tasks.

Ferrari *et al.* [65] presented a pose estimation for 2D video sequences. They initialized the upper body position with a detector based on Histograms of Oriented Gradients [57]. Then, they limited the positions of body parts by using a GrabCut segmentation [136]. After estimating the first pose, they used both the appearance of the body parts as well as the joint angles for tracking. The problem with such multi-stage approaches is that there is potential for failure in each step. Further, the initialized appearance model is sensitive to changing conditions. For evaluation, Ferrari *et al.* [65] used a score that is often used in the context of pose estimation with pictorial structures: the percentage of correct parts (PCP). A part is considered correct if both estimated end joints are within a distance of at most half of the limb size. Their reported average PCP score for the upper body was 56.

Another application of pictorial structures to pose estimation in 2D was presented by Andriluka *et al.* [29]. They learned both the part appearances as well as the connection prior from training data. In particular, their appearance model is strongly specialized for the various body parts to compute part evidence maps. Their reported PCP scores for the upper body on two different datasets are 55.2 and 73.5. As with all training-based approaches, they introduce an implicit bias through the training data.

In another work, Amin *et al.* [26] used pictorial structures for multi-view pose estimation. The unary part evaluation term is based on learned image features and connections between parts are evaluated using learned Gaussian distributions. The first step is estimating the 2D pose in each single image. The 3D pose is then estimated by triangulating the 2D joint positions in 3D. Using all available constraints and features, they achieve an average joint position error of 54.5 mm on selected scenes of the HumanEva-I dataset [148].

Pictorial structures have mostly been used for 2D pose estimation. The reason is that the combination of parts is even more complex in 3D. Burenius *et al.* [45] addressed this problem by proposing a 3D pictorial structures approach. The unary appearance term is evaluated by using weakly trained part detectors. To compute the binary connection term, they enforced three kinds of constraints. First, skeleton constraints to fix the limb sizes and to avoid a tolerance for the joint positions. Second, view constraints that limit the positions of parts to locations where the part detector found sufficient evidence. And third, joint angle constraints by limiting possible rotations to a uniformly sampled space of fixed rotations. Their reported PCP scores on a custom dataset with two different settings are 70 and 77. While they showed a significant complexity reduction, this reduction is mostly due to a discretization of the search space. Burenius *et al.* [45] is one of the few works that also report runtimes. There, the reported runtimes are between 1 second and 69 minutes depending on the granularity of the translation and rotation spaces.

The algorithms discussed above generally rely on trained part detectors to limit the search space. In contrast, the algorithm presented in this work shows how segmentation as a preprocessing step can be used to effectively reduce the search space. In addition, this reduction is meaningful in that it respects the observed data and adequately represents body parts. Further, the approach presented in Section 4 achieves real-time runtimes.

3 Volume and Super Segmentations

This chapter presents the contributions on segmentation methods in this thesis with a focus on computational efficiency through a reduction of the number of elements. In the context of this work, segmentation serves two purposes. First, it provides a unique representation for various sensor types through volumetric segmentation and reconstruction. Second, through super segmentation techniques, the number of inputs and, therefore, the search space for following processing steps is reduced.

First, Section 3.1 introduces a real-time voxel carving algorithm that is designed to compute a volumetric reconstruction of the environment with both video cameras as well as depth sensors. For video cameras, it also addresses the challenge of static occlusions. For depth sensors, it provides a way to compute volumetric data even with only one sensor. Then, super segmentation techniques are introduced. Section 3.2 presents a superpixel algorithm with a focus on compactness. This algorithm is the basis for 3D supervoxel segmentation presented in Section 3.3 that works on voxels and that provides the building blocks for pose estimation.

3.1 Voxel Carving

This section introduces the voxel carving algorithm developed in this thesis. Voxel carving provides a 3D reconstruction of the observed scene represented by voxels, small volumetric cubes that act as atomic units, similar to pixels of an image. The algorithm presented here combines efficiency and robustness with flexibility. In addition, it provides a solution to deal with static occlusions by introducing 3D occlusion maps. In addition, it is flexible because it can work with various sensor types for different environments, *e.g.*, video cameras and depth sensors, and with a variable number of sensors. This allows for a variety of supported environments where this voxel carving algorithm can be used. Because the voxel carving is the first step of the system presented in this work, this flexibility carries over to all following steps, in particular pose estimation.

3.1.1 General Voxel Carving

The proposed voxel carving algorithm approximates a 3D reconstruction of a scene with voxels. The scene is observed with multiple calibrated cameras. The volume of interest is given by the user and fixed. Section 2.1.1 provides details on camera calibration and projection functions.

A voxel v is a solid 3D cube with fixed side length. The set of voxels $\mathfrak{V}$, the voxel grid, partitions the given volume of interest. The voxel cell of voxel $v \in \mathfrak{V}$ is occupied ($v = 1$) in case it belongs to the visual hull and empty ($v = 0$) if the corresponding voxel is carved. Let C be the set of calibrated cameras and $\mathcal{I}_c$ the corresponding image of camera $c \in C$. $\mathcal{I}_c(u)$ gives the pixel value of image $\mathcal{I}_c$ at coordinates u. The function $p(v, c)$ projects the center of voxel v to image coordinates of camera c. Further, let the general carving function $\delta_c(\mathcal{I}_c, v)$ be true if image $\mathcal{I}_c$ does not support the existence of voxel v, *e.g.*, because the voxel projects into the background. Then, a voxel only remains in the voxel grid, *i.e.*, is not carved, if the carving function is false for all images.

The occupancy for each voxel is then given by

$$v = \begin{cases} 1 & \text{if } \forall c \in C : \delta_c(\mathcal{I}_c, v) = false \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

Algorithm 1 shows pseudo-code for an implementation of the generic voxel carving algorithm. Adapting this algorithm to specific use cases, *e.g.*, multi-view video or depth images with and without occlusion maps, requires only a modification of the carving function δ. Similar to [89], it is possible to relax the carving function by allowing a fixed number of silhouette misses before carving the voxel, but this is not required in this work.

Multi-View Video

This section describes the application of voxel carving to multi-view videos. For each image $\mathcal{I}$, a foreground segmentation F is computed. Details of foreground segmentation can be found in Section 2.1.2. A voxel is carved if it projects outside the silhouette and into the background for at least one image. The carving function $\delta_c(F_c, v)$ then is

$$\delta_c(F_c, v) = \begin{cases} true & \text{if } F_c(p(v, c)) = 0 \\ false & \text{otherwise} \end{cases} \tag{3.2}$$

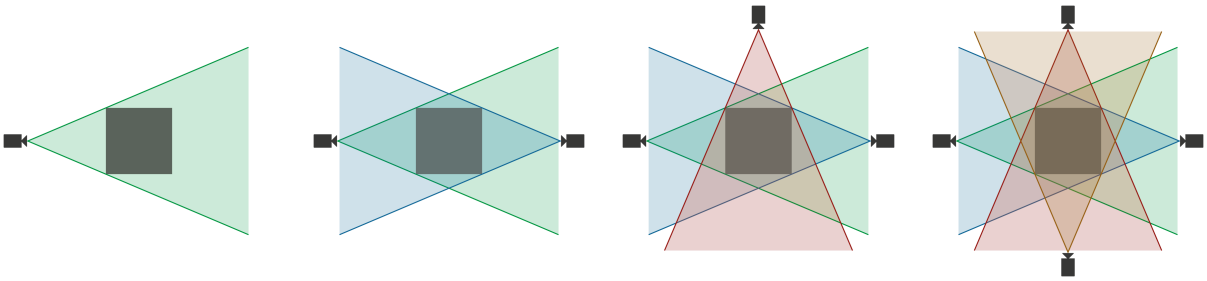
Algorithm 1 Generic Voxel Carving

```
1: Input: camera configurations C
2: Input: ∀c ∈ C : load I_c
3: Output: voxel grid 𝔙
4:
5: Initialize voxel grid 𝔙
6: for all Voxel v ∈ 𝔙 do
7:    v ← 1
8:    for all Camera c ∈ C do
9:       if δ_c(I_c, v) = true then
10:         v ← 0
11:      end if
12:   end for
13: end for
```

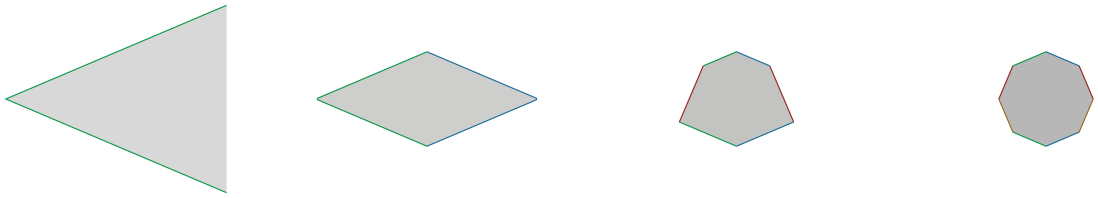
Figure 3.1 illustrates reconstructed volumes for an increasing number of cameras. As described in Section 2.1, voxel carving cannot reconstruct concave surfaces. In addition, the reconstruction can be too coarse when too few cameras are used (Section 2.1.3). Another challenge are shadow effects or *hallucinated* visual hulls when more than one object are present. The reason is that the rays cast for volumetric carving can intersect more than once when multiple objects are present. Because the silhouettes only represent binary information, it cannot be decided which intersection is the correct one. This leads to additional reconstructed volumes as Figure 3.2 shows. These problems are not as severe when using depth sensors as will be discussed in the next section.

Depth Images

This section describes the application of voxel carving to depth images for multi-view and single-view setups. Voxel carving with multi-view depth images is a direct extension of the multi-view video case. In contrast to binary silhouette images, the pixels of depth images D give the distance of the closest object to the camera at their position. This additional information can be used to improve the computation of the visual hull by moving the reconstructed surface closer to the actual surface of the object. Figure 3.1 shows reconstructed volumes with depth cameras. By using depth information, it is also possible to reconstruct observed concave surfaces as Figure 3.3 shows. In addition, the problem of hallucinated visual hulls, as discussed above, is not as severe as for the multi-view video setup as Figure 3.2 shows.



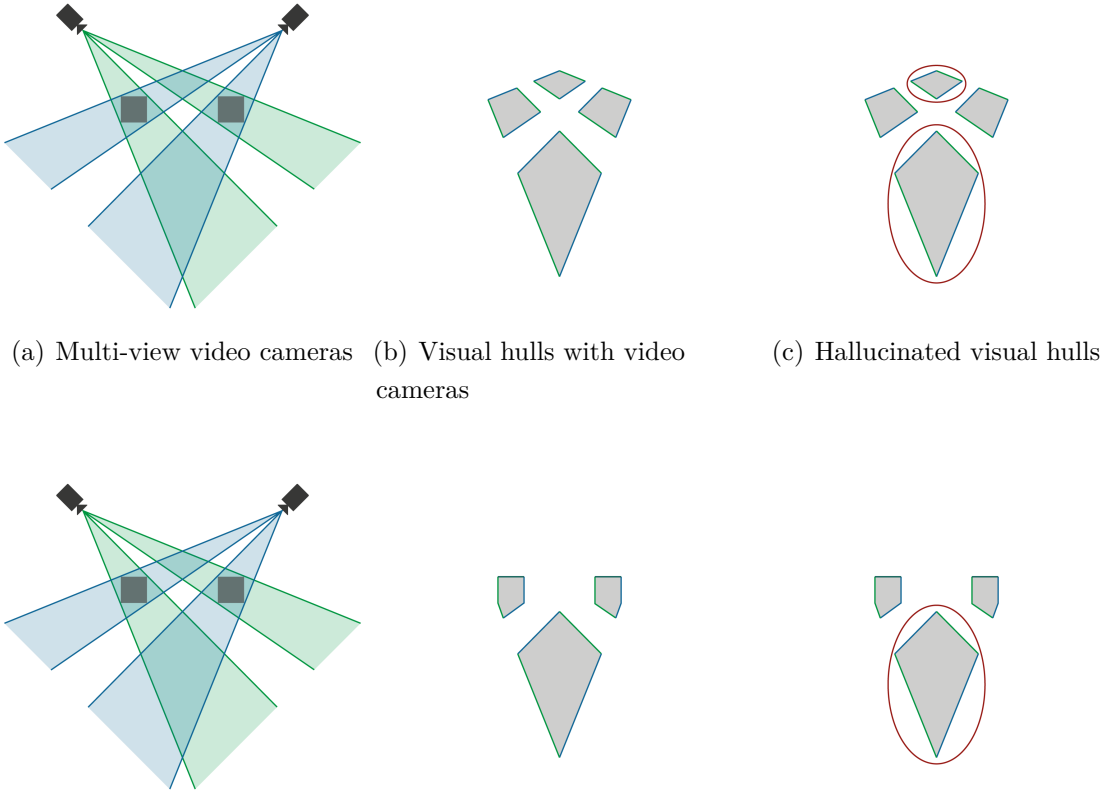
(a) Sensor setup



(b) Voxel carving with video cameras

(c) Voxel carving with depth sensors

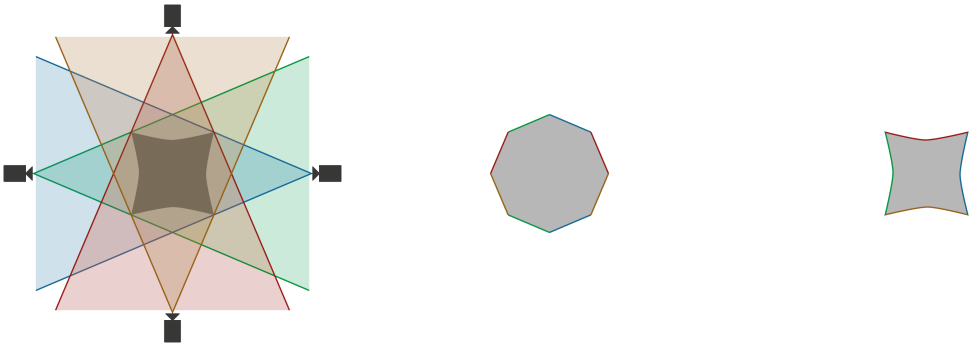
Figure 3.1: Voxel carving for multi-view video and depth sensors with an increasing number of cameras. (a) shows the camera configurations. (b) shows results for voxel carving for multi-view video. Reconstruction accuracy increases with more cameras, but remains an overestimation of the original object. (c) shows voxel carving with multi-view depth sensors. By using the depth information in addition to the intersections, the visual hull boundaries are closer to the object boundaries.



(a) Multi-view video cameras (b) Visual hulls with video cameras (c) Hallucinated visual hulls

(d) Multi-view depth sensors (e) Visual hulls with depth sensors (f) Hallucinated visual hulls

Figure 3.2: Hallucinated visual hulls. Too few cameras and poor camera placement leads to the reconstruction of volumes that are not supported by real objects. This effect occurs due to multiple intersections of camera rays during the volume intersection. This effect is more severe for multi-view video cameras. In case of depth sensors, the depth information helps to avoid hallucinated visual hulls in front of objects. Occluded areas, however, are still affected by this effect.



(a) Sensor setup (b) Video cameras (c) Depth sensors

Figure 3.3: Voxel carving of objects with concavities. (a) shows the placement of the video cameras or depth sensors. (b) illustrates that concave surfaces cannot be reconstructed with video cameras whereas (c) shows that this is possible with the additional information provided by depth sensors.

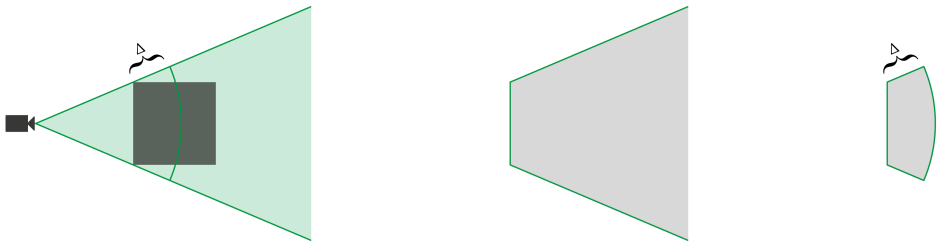
The additional information given by the depth images D can be used in the carving function $\delta_c(D_c, v)$ as follows. Let $d(v, c)$ give the distance of voxel v to camera c. A voxel is carved if the observed distance at the corresponding pixel in at least one depth image is greater than the distance of this voxel to the camera, *i.e.*, the voxel is in front of the closest object in at least one view. Thus, the carving function for depth images is

$$\delta_c(D_c, v) = \begin{cases} true & \text{if } d(v,c) < D_c(p(v,c)) \\ false & \text{otherwise} \end{cases} \tag{3.3}$$

Single-View Depth Image

Depth sensors give a 2.5D reconstruction by providing the depth maps of observed surfaces. The surface points can be converted to 3D, but the reconstruction still only considers the visible surface. As explained above, multi-view depth carving gives a volumetric 3D reconstruction. However, there are cases where only one depth sensor is available, but a volumetric reconstruction is required. An example is the pose estimation algorithm presented in this work that uses the additional information provided by the occupied volumes and therefore requires volumetric information.

Accurate volumetric reconstruction is not possible with only one depth sensor. However, it can be approximated by limiting the penetration depth, *i.e.*, the maximum allowed



(a) One depth sensor (b) Original voxel carving (c) Voxel carving with penetration threshold Δ

Figure 3.4: Voxel carving with a single depth camera. (a) Only one depth sensor is available which leads to (b) a severe overestimation of the volume. (c) By limiting the penetration to a fixed depth with threshold Δ, the size of the reconstructed volume can be limited. This is particularly useful if the expected depth of the object is approximately known.

distance Δ of voxels to their closest surface point along the camera rays. The modified carving function then is

$$\delta_c(D_c, v) = \begin{cases} true & \text{if } d(v,c) < D_c(p(v,c)) \vee \\ & \quad D_c(p(v,c)) + \Delta < d(v,c) \\ false & \text{otherwise} \end{cases} \tag{3.4}$$

Figure 3.4 gives an example of an approximate volumetric reconstruction with penetration threshold Δ.

3.1.2 Occlusion maps

For voxel carving with multi-view videos, occlusions caused by static objects are a common source of errors. Static occluders typically become part of background models used in foreground segmentations which leads to missing foreground parts in the silhouettes of objects. Due to the nature of voxel carving, an error in only one silhouette is sufficient to cause severe defects in the visual hull.

The discussion of related work in Section 2.1 showed that methods to handle static occlusions either do not result in minimal visual hulls or are computationally too expensive. The

solution which was first published in [6] gives a minimal visual hull without imposing severe computational overhead:

Let O be the set of occlusion maps for foreground segmentations F. The occlusion maps store for each pixel the distance of the closest occluder to the camera along the ray passing through it or infinite in case of no occlusion. In fact, the occlusion maps are similar to images of depth sensors as discussed above. The occlusion maps can either be manually annotated by defining geometric objects and projecting them into the image or generated by scanning the environment with depth sensors. Given the occlusion maps, the modified carving function is then given by

$$\delta_c(F_c, O_c, v) = \begin{cases} true & \text{if } F_c(p(v,c)) = 0 \wedge \\ & \quad d(v,c) < O_c(p(v,c)) \\ false & \text{otherwise} \end{cases} \tag{3.5}$$

This solution is very efficient. It only requires the computation of the distance between voxels and cameras and an additional lookup in the precomputed occlusion maps. It effectively cancels the effects of static occlusions by not carving occluded voxels. And it results in minimal visual hulls because the carving function is only relaxed for voxels behind occluders. In contrast, approaches with binary occlusion maps would also keep voxels in front of occluders resulting in larger visual hulls.

With the modified carving function, voxels that are occluded in all views are not carved. Consequently, voxels inside occluding objects are not removed. To avoid this effect, the voxel carving algorithm is modified to ensure that there is at least one true silhouette hit for existing voxels, *i.e.*, they are in front of an occluding object in at least one view. An example of occlusion maps are shown in Section 3.1.4 in Figure 3.9. There, an example image with static occlusion and the corresponding occlusion map together with voxel carving results with and without occlusion maps are shown.

3.1.3 Computational Complexity

For human-computer interaction, real-time processing is crucial. For a good user experience, the system reaction should be below 50 ms, as stated by Von Hardenberg and Bérard [167] based on Card *et al.* [49]. In this work, voxel carving is one of the first preprocessing steps. Therefore, it has to be computationally efficient to not act as a bottle neck. This section describes the design considerations for the proposed voxel carving algorithm. The algorithm design favors parallel execution on the GPU of a graphics board.

Minimizing Projection Computations

Voxels are volumetric objects and, if they are close enough to the camera, do not project on one single pixel. Therefore, all eight voxel corners must be projected into the image to get the correct bounding area. Even when using integral images [165], this is very inefficient and requires about one order of magnitude more computations. Ladikos *et al.* [91] circumvent this problem by scaling the foreground segmentations so that each voxel projects only on one pixel. This, however, is only an approximate solution.

The voxel carving algorithm used in this work projects only the centers of voxels with function $p(v, c)$. This can potentially lead to an overestimation of the visual hull. For example, if the voxel center projects directly on a silhouette edge, parts of the voxel are actually outside the object, thus leading to an overestimation. However, when considering the full 2D projection area, this voxel would be carved resulting in an underestimation of the reconstructed volume. This implies that both approaches are only approximations due to the space discretization related to the voxel size. Therefore, this work uses the computationally less expensive approach and only projects voxel centers. It tolerates a certain overestimation of reconstructed volumes and offers finer precision by adjusting the voxel size. The accuracy of projecting all eight voxel corners can then be approximated by using voxels with half the original side length.

Parallel Design

Efficient GPU implementations favor computations over memory access and conditional statements. On the GPU, it is often beneficial for the overall runtime to prefer redundant computations over storing results in memory. Due to the increasing power of parallel processors on off-the-shelf graphics boards, GPU implementations have proven advantageous for real-time applications [91], as is the case in this work.

Algorithm 1 is designed to follow a very linear execution structure. It has only one conditional statement that is very deep within the program flow chart. The decision to project only voxel centers and to not use an octree representation supports this design because it eliminates additional conditional statements.

The operations used for the projection in Algorithm 1 are the same for all cameras and fall into the single instruction, multiple data (SIMD) class of algorithms that are well suited for execution on the GPU. In addition, due to the spatial proximity of neighboring voxels in the grid, their projections are also close in image space which is beneficial for cached memory lookups.

Complexity Analysis

This section analyzes the time and space complexity of the proposed voxel carving algorithm shown in Algorithm 1 and its variations for multi-view video and depth sensors with and without occlusion maps.

The time complexity is linear in the number of voxels and cameras. Let $|\mathfrak{V}|$ give the number voxels and $|C|$ the number of cameras. In the worst case, each voxel has to be projected into every camera image resulting in a time complexity in $\mathcal{O}(|\mathfrak{V}| \cdot |C|)$. The time complexity remains the same for all variations of this algorithm introduced above, even with the addition of occlusion maps. This is because occlusion map lookups are constant in the number of cameras and do not change the complexity class.

Regarding the worst case for space complexity, this algorithm requires storing the full voxel grid of size $|\mathfrak{V}|$ and all camera images. Let $|\mathcal{I}|$ be the number of entries of an image, either a foreground segmentation or a depth image. The space complexity is then in $\mathcal{O}(|\mathfrak{V}| + |C| \cdot |\mathcal{I}|)$. Again, the space complexity remains the same for all variations.

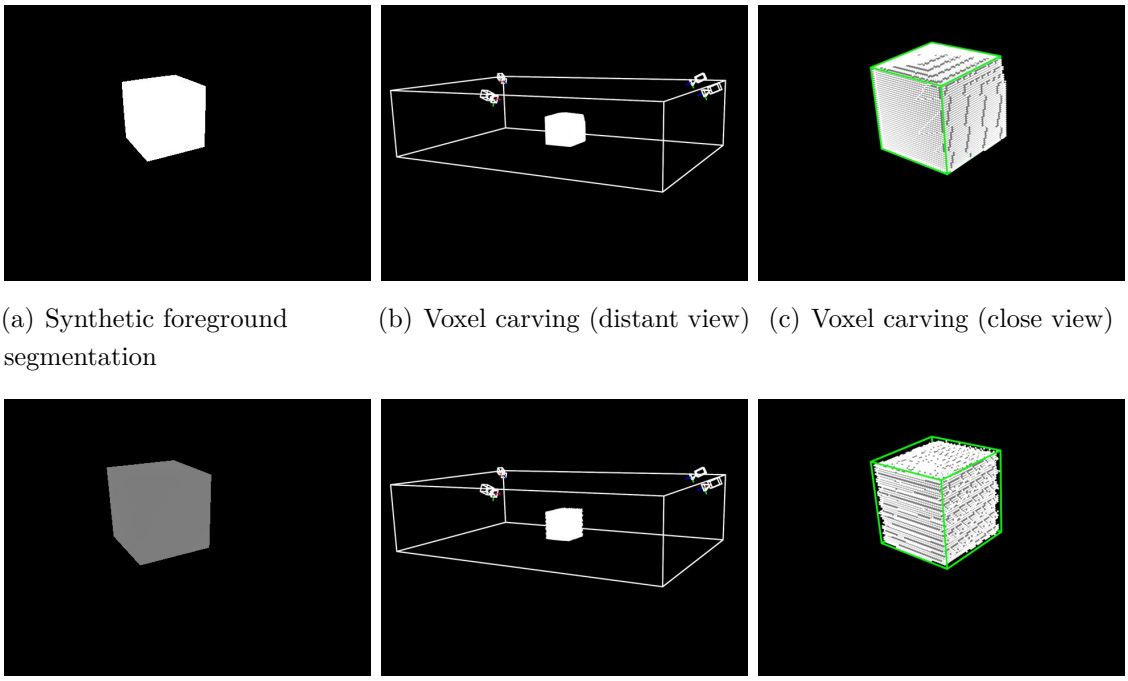
3.1.4 Evaluation

This section provides an evaluation of the voxel carving algorithm with both synthetic as well as real-world examples. The evaluation includes qualitative results as well as a runtime analysis. It uses the camera matrices and room definition provided by the UMPM dataset [162] that is also used for pose estimation in Section 4.

Synthetic Examples

This section shows synthetic examples for multi-view video and depth sensors. Given the camera matrices, objects are rendered into the images to simulate observations. The scenario in Figure 3.5 shows a box with the dimensions 1 $m \times 1$ $m \times 1$ m placed at the room center with an accurate approximation of the box volume for both multi-view video as well as depth images. Figure 3.6 shows the shadow effects that can occur during voxel carving without a sufficient number of cameras and poor camera placement.

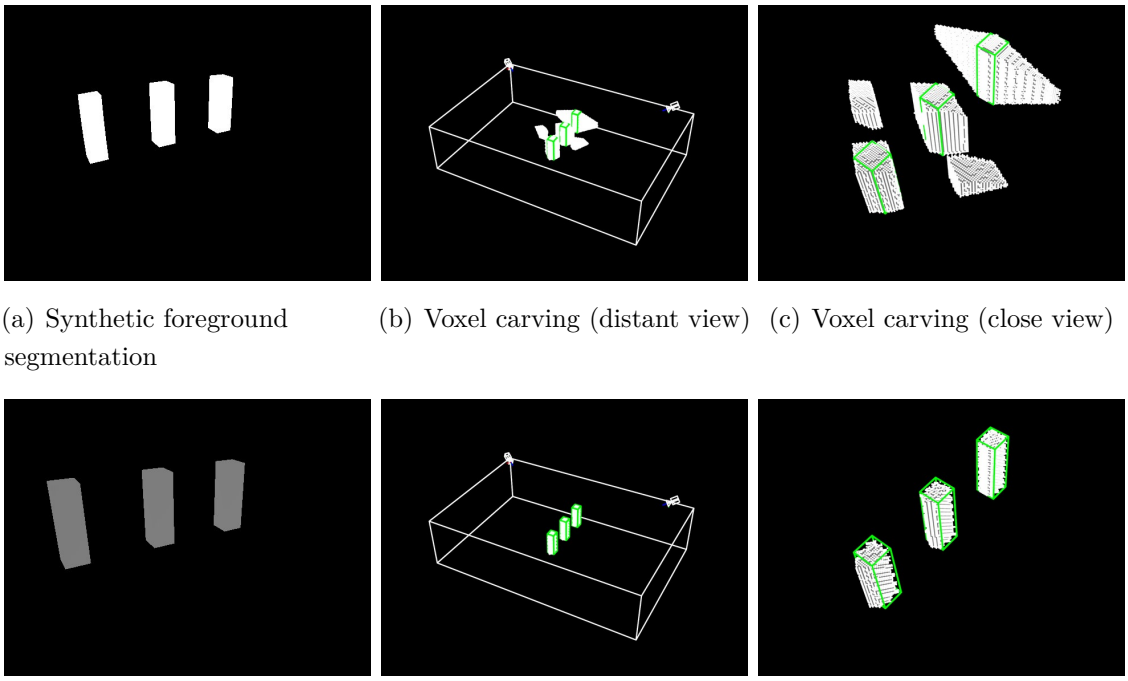
The examples show that a sufficient number of cameras should be placed with opposing views for good voxel carving results. This is the case for the datasets used in this work where at least four cameras are used that are placed near the room corners.



(a) Synthetic foreground segmentation (b) Voxel carving (distant view) (c) Voxel carving (close view)

(d) Synthetic depth rendering (e) Voxel carving (distant view) (f) Voxel carving (close view)

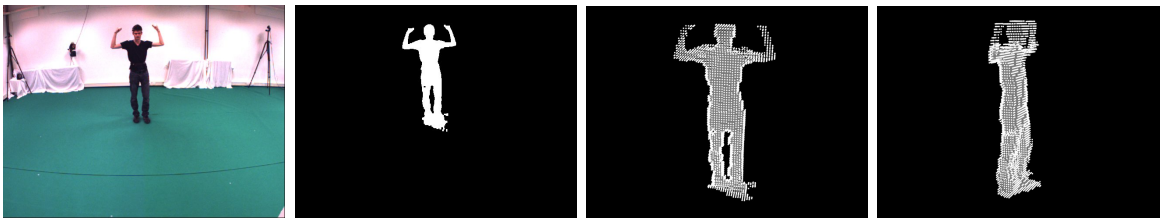
Figure 3.5: Voxel carving with synthetic images. The 1 $m \times 1$ $m \times 1$ m box at the room center is rendered as foreground segmentations (top row) and as depth images (bottom row) with a voxel size of 2.5 cm. (a+d) show one of the four camera views, (b+e) the voxel carving results including the camera setup from a distance and (c+f) from a close view with the edges of the box shown in green. (c) shows that the reconstruction is slightly convex due to the volume intersection and that it extends outside the original box whereas (f) shows an underestimation of the box volume due to depth discretization and because the algorithm only projects the centers of the voxels.



(a) Synthetic foreground segmentation (b) Voxel carving (distant view) (c) Voxel carving (close view)

(d) Synthetic depth rendering (e) Voxel carving (distant view) (f) Voxel carving (close view)

Figure 3.6: Voxel carving with synthetic images, showing too few cameras and poor camera placement. Three pillars with dimensions 0.3 $m \times 0.3$ $m \times 1.0$ m are placed 1 m apart along the x-axis. The top row shows results for foreground segmentations and the bottom row for depth images. The voxel size is 2.5 cm. The green lines represent the ground truth pillar edges. Using only two cameras that are mounted to the same side of the room, leads to very poor voxel carving results with foreground images. The top row shows shadow or hallucinated visual hulls as well as large overestimations of the volumes. With depth information (bottom row), the reconstruction quality is strongly improved.



(a) Camera image (b) Foreground segmentation (c) Voxel carving (frontal view) (d) Voxel carving (side view)

Figure 3.7: Voxel carving with multi-view video. (a) shows one of the four views from a sequence of the UMPM dataset [162] together with the foreground segmentation in (b). A frontal view of the 3D reconstruction is shown in (c) and a side view in (d).

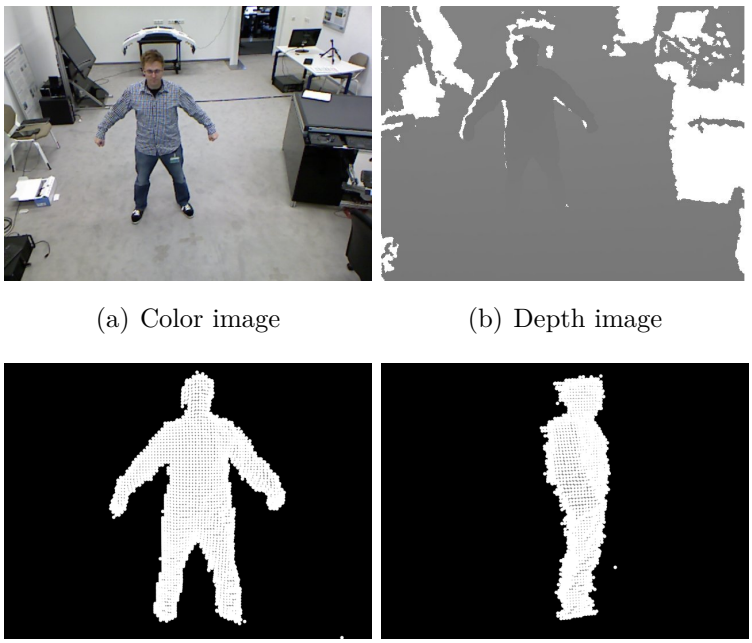
Real-World Examples

This section shows voxel carving results for recorded video sequences. For multi-view voxel carving, four camera views from the UMPM dataset [162] are used. For voxel carving with depth images, recorded views of one Kinect camera are used.

Real-world examples are shown in Figures 3.7 and 3.8. Figure 3.7 shows voxel carving results with data from the UMPM dataset [162]. The voxel carving is based on foreground segmentations for four camera views. The reconstructed visual hull corresponds to the shape of the person. Figure 3.8 shows voxel carving results with one Microsoft Kinect depth sensor. The reconstruction shows that a similar performance is possible with fewer cameras given the additional depth information.

Occlusion Maps

This section shows voxel carving results in the presence of static occlusions with and without occlusion maps. For the UMPM dataset [162], the occlusion maps have been generated manually by defining the 3D coordinates of the occluding objects. Figure 3.9 shows an example with and without an occlusion map under the presence of heavy occlusions. Without occlusion maps, there are severe defects in the visual hull and large parts of the legs are removed. With occlusion maps, the legs are correctly reconstructed and are part of the visual hull. However, the relaxation of the voxel carving function also leads to hallucinated voxels near to the occluding object.

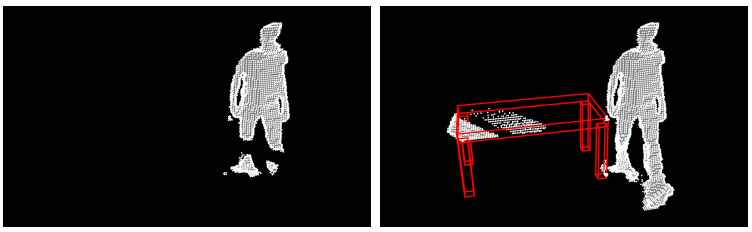


(a) Color image (b) Depth image

(c) Frontal view (d) Side view

Figure 3.8: Voxel carving with a single depth camera. The images show that accurate voxel carving results can be computed even with only a single depth camera. Here, the penetration depth was set to 20 cm.

(a) Camera image (b) Foreground segmentation (c) Occlusion map



(d) Voxel carving without occlusion map (e) Voxel carving with occlusion map

Figure 3.9: Voxel carving with occlusion maps. This sequences includes occlusions by a table. The red lines show the table edges. (a) shows one camera view and (b) the corresponding foreground segmentation. Occluded regions are not part of the foreground. (c) shows the occlusion map. (d) shows voxel carving without occlusion maps. There are severe defects in the visual hull. (e) shows results with occlusion maps. The legs are part of the visual hull at the cost of hallucinated voxels close to the occluding object.

Runtime Evaluation

This section gives quantitative runtime results for the voxel carving algorithm. The example images used for runtime measurements are the synthetic box images introduced in Section 3.1.4 as well as worst and best case images. For better comparison to other work, the number of voxels was set to 64^3, 128^3, and 256^3. With room dimensions of 6.0 m × 9.2 m × 2.2 m, this resulted in voxel sizes of approximately 7.6 cm, 3.8 cm, and 1.9 cm.

The evaluation included three scenarios: 1) the cube from the synthetic examples at the beginning of this section in order to represent a normal scenario, 2) best case scenario with black images where all voxels are removed (*i.e.*, all background), and 3) worst case scenario with white images where all voxels remain (*i.e.*, all foreground). The runtimes were measured for one to four cameras. The experimental system was an Intel Pentium Intel(R) Core(TM) i7-3770 CPU with 3.40 GHz and a NVIDIA GeForce GTX 660 Ti. Tables 3.1 and 3.2 show measurements for both multi-view video and depth sensors.

For the worst case scenario, the runtimes increase with the number of cameras. The reason is that in the worst case scenario, all voxels must be evaluated in all cameras. This is not the case for the other scenarios where the algorithm stops evaluating a voxel for the remaining cameras, as soon as there is evidence that it should be carved. In typical scenarios, most of the image is background and voxels get carved relatively soon. Therefore, the runtimes for the normal and best case scenario are very close and do increase only slightly with the number of cameras. A comparison of voxel numbers to runtimes shows that the algorithm scales linearly in the number of voxels. Voxel carving with multi-view video is slightly faster than depth carving.

When comparing the results to the algorithms proposed by Ladikos *et al.* [91], the different graphics boards must be considered. In [91], a NVIDIA 8800 GTX was used that has a significantly lower number of cores (128 compared to 1344, approximately one tenth). The algorithm most comparable to this work, *GPU2*, achieves runtimes of 18.60 ms, 113.88 ms, and 870.06 ms for 64^3, 128^3, and 256^3 voxels, respectively and is approximately 15 to 40 times slower. They used 16 cameras, but distributed the algorithm across 4 PCs which evens out the workload for each PC. Their optimized version, *GPU2_OT*, achieves runtimes of 15.24 ms, 25.91 ms, and 73.53 ms is still about 15 times slower for lower voxel numbers, but scales better for larger numbers. However, this algorithm uses precomputed rescaled silhouette images to imitate octree behavior. While this is possible for binary silhouette images, it cannot be applied in the same way to depth images. Therefore, the presented algorithm is not only faster, but also more flexible than [91].

Table 3.1: Voxel carving runtimes for multi-view video. The table shows runtimes for varying voxel sizes and resolutions and for different numbers of cameras. The best and worst case runtimes are the result of empty voxel grids for black images (*i.e.*, only background) and full voxel grids for white images (*i.e.*, only foreground).

Voxels [*Number*]	Voxel size [*cm*]	Cameras [*Number*]	Synthetic cube [*ms*]		Best case [*ms*]		Worst case [*ms*]	
			CPU	GPU	CPU	GPU	CPU	GPU
64^3	7.6	1	17.8	0.7	17.9	0.7	18.0	0.7
64^3	7.6	2	19.5	0.8	17.8	0.8	26.8	0.9
64^3	7.6	3	18.6	0.9	17.8	0.9	34.4	1.1
64^3	7.6	4	19.5	1.1	17.8	1.0	42.5	1.3
128^3	3.8	1	145.1	3.0	145.1	3.1	152.0	3.1
128^3	3.8	2	158.8	3.3	156.5	3.1	215.0	3.8
128^3	3.8	3	152.0	3.4	145.1	3.2	281.4	4.6
128^3	3.8	4	153.3	3.6	151.3	3.3	345.4	5.3
256^3	1.9	1	1178.5	20.0	1171.8	19.1	1191.9	20.0
256^3	1.9	2	1213.2	19.6	1174.8	20.2	1818.7	22.5
256^3	1.9	3	1225.8	19.9	1172.8	20.3	2238.7	26.3
256^3	1.9	4	1239.8	20.3	1225.7	22.1	2772.4	30.0

3.1.5 Conclusion

This section introduced the voxel carving algorithm developed in this work. It is computationally very efficient which is necessary for real-time human-computer interaction applications. It circumvents defects caused by static occlusions with occlusion maps.

Further, the algorithm is designed to work with interchangeable carving functions for different sensor types. This introduces flexibility for algorithms working on top of the voxel representation. They can be used without modifications in various environments and with different sensor setups by choosing an appropriate carving function.

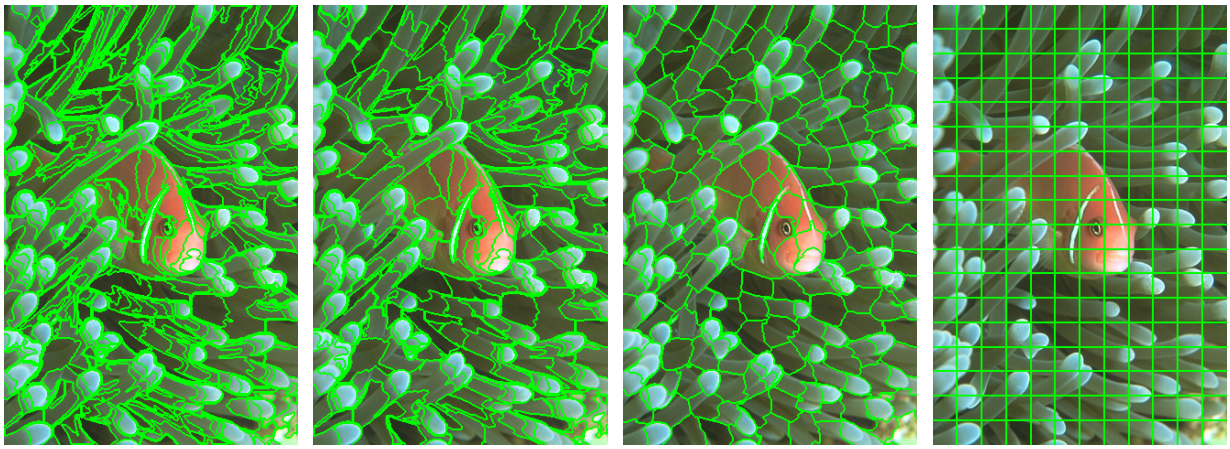
Table 3.2: Voxel carving runtimes for multi-view depth sensors. The tables shows runtimes for varying voxel sizes and resolutions and for different numbers of cameras. The best and worst case runtimes are the result of empty voxel grids for images without depth values and full voxel grids for images that simulate objects right in front of the camera lenses.

Voxels [*Number*]	Voxel size [*cm*]	Cameras [*Number*]	Synthetic cube [*ms*]		Best case [*ms*]		Worst case [*ms*]	
			CPU	GPU	CPU	GPU	CPU	GPU
64^3	7.6	1	16.7	1.0	16.5	1.0	17.0	1.0
64^3	7.6	2	17.2	1.2	16.6	1.2	22.9	1.3
64^3	7.6	3	17.3	1.4	16.6	1.3	28.3	1.6
64^3	7.6	4	17.5	1.6	20.3	1.5	35.9	2.3
128^3	3.8	1	135.4	3.7	136.6	3.6	138.8	3.8
128^3	3.8	2	142.2	4.0	135.5	3.9	184.3	5.0
128^3	3.8	3	140.1	4.3	135.5	4.0	226.3	6.1
128^3	3.8	4	141.3	4.4	165.1	4.2	286.4	7.0
256^3	1.9	1	1092.7	25.0	1090.8	21.6	1119.5	23.7
256^3	1.9	2	1118.6	22.6	1091.3	23.1	1481.4	28.7
256^3	1.9	3	1129.2	23.2	1091.2	22.0	1813.6	34.5
256^3	1.9	4	1177.5	23.8	1334.4	23.4	2312.7	39.7

3.2 Superpixel Segmentation

Superpixel segmentation is an oversegmentation technique that partitions an image into a large number of connected segments, the superpixels, based on a similarity measure and a grouping criterion. The main purpose of superpixels is a reduction of the number of pixels and, consequently, the number of input elements for following processing steps. An overview about superpixels and superpixel segmentation algorithms was given in Section 2.2.1.

This section introduces the superpixel segmentation algorithm and the metric to measure the compactness of a given segmentation that were developed during this thesis. The superpixel segmentation is the basis of the supervoxel segmentation that later provides the building blocks for 3D pose estimation. In the first part of this section, the concept of superpixel compactness and a metric to measure the compactness of a given superpixel segmentation are introduced. The second part of this section presents the superpixel algorithm and a comparison to other segmentation algorithms.



(a) $\alpha = 0$ (b) $\alpha = 0.9$ (c) $\alpha = 0.98$ (d) $\alpha = 1$

Figure 3.10: Superpixel segmentations with various degrees of compactness. The compactness parameter α increases from no emphasis on compactness (left) to full compactness (right). The most compact segmentation is actually identical to a grid.

3.2.1 Compactness

Compactness is a measure of shape. As Section 2.2.1 showed, there is an agreement between authors of superpixel segmentation algorithms that compactness is a desirable property [19, 95, 97, 111, 123, 163, 183, 185]. For visual reference, Figure 3.10 shows examples of superpixel segmentations with varying degrees of compactness.

Compactness is not only preferable from an aesthetic point of view. Compact superpixels are a better representation of spatially coherent information than superpixels that stretch across the image. Also, it is computationally less expensive to extract information from compact superpixels with regular boundaries simply because the overall boundary length is shorter and easier to traverse. Further, the size of the superpixels defines the level of detail they should capture. Non-compact superpixels with highly irregular boundaries that try to capture every minor image detail can then be compared to overfitting in machine learning.

In mathematics, measuring the compactness of a shape is a well-known task that is related to the isoperimetric problem: for a given shape with boundary length L, find the 2D shape with the same boundary length that maximizes the area [128]. In 2D, the solution is the circle. The compactness of a shape can then be quantified with the isoperimetric quotient that is given by the area of the shape divided by the area of a circle with the same boundary length.

The notion of the isoperimetric quotient can be directly applied to measure the compactness of a superpixel. Let A_S be the area A of superpixel S and let L_S be its boundary length. The radius of the circle with the same boundary length is then given by $r = \frac{L_S}{2\pi}$. With A_C being the area of a circle with radius r, the isoperimetric quotient Q of superpixel S is

$$Q_S = \frac{A_S}{A_c} = \frac{4\pi A_S}{L_S^2}. \tag{3.6}$$

The compactness of a superpixel segmentation represented by a set of superpixels $\mathcal{S}$ is then given by the sum of the isoperimetric quotients of all superpixels $S \in \mathcal{S}$ normalized with their size $|S|$ relative to the full segmentation size $|\mathcal{S}|$. Normalizing with the superpixel size instead of the number of superpixels is important to avoid that many small, but perfectly compact, superpixels dominate the overall score. The compactness CO of superpixel segmentation $\mathcal{S}$ then is

$$CO(\mathcal{S}) = \sum_{S \in \mathcal{S}} \Big(Q_S \cdot \frac{|S|}{|\mathcal{S}|} \Big). \tag{3.7}$$

This compactness measure is used to quantify the compactness of superpixel segmentations. The next section introduces an algorithm that offers a transparent way to control the superpixel compactness. An evaluation to other superpixel segmentation algorithms and an investigation of the impact of compactness on the segmentation quality is presented in Section 3.2.3.

3.2.2 Superpixel Segmentation of Images

The superpixel segmentation algorithm presented here is motivated by SLIC [19]. SLIC is an iterative algorithm that groups pixels with k-means clustering using superpixel centers as seeds. Its similarity measure combines a weighted sum of color differences in Lab color space and Euclidean distances in pixels. However, only the Euclidean distance term is weighted. Further, due to this clustering, the connectivity of pixels cannot be guaranteed and superpixels can be ripped apart. This requires a final postprocessing step to restore connectivity.

The algorithm presented here overcomes these drawbacks by weighting both distance terms in a transparent way. Also, connectivity is maintained during segmentation by applying the similarity measure only to boundary pixels. In this way, the superpixels remain connected

and cannot be ripped apart. Additionally, the algorithm is very efficient because the similarity measure is only applied to boundary pixels.

Algorithm

The superpixel segmentation algorithm developed in this work is an iterative algorithm that refines the boundaries of an initial superpixel segmentation by evolving them towards the final segmentation. The algorithm requires only two parameters: the size of the initial superpixels and the compactness weight α that controls the superpixel compactness. One key aspect is that all operations are strictly limited to boundary pixels and only require information of the immediate neighborhood around them. The advantage of this local design will be discussed later.

The segmentation is initialized with a rectangular grid of superpixels that is given by the superpixel size as Figure 3.11a shows. Then, in each iteration, the similarity of every boundary pixel to each superpixel in its immediate 4-neighborhood is computed. Then, the pixel is assigned to the most similar superpixel. This is repeated for a given number of iterations or until the segmentation converges (Figure 3.11d). By iteratively modifying the boundaries, the algorithm can ensure that the superpixels remain connected and are not ripped apart during segmentation as will be shown after introducing the algorithm.

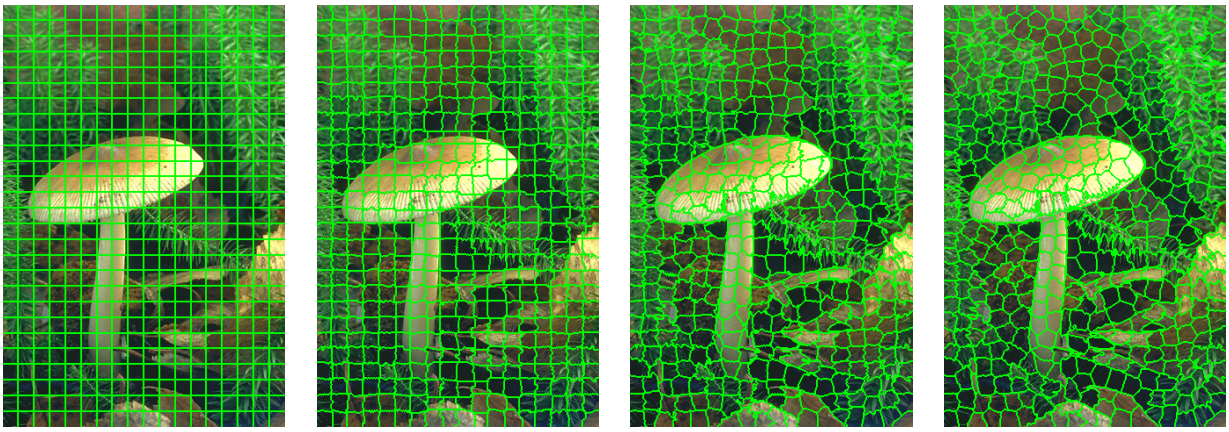
The similarity between a pixel p and a superpixel S is expressed by the dissimilarity term ψ. The more similar a pixel is to a superpixel, the smaller is the corresponding dissimilarity measure ψ. The term ψ combines the color space distance $d_{rgb}(p, S)$ in RGB color space with the Euclidean distance $d_{xy}(p, S)$ measured in pixels. Both terms are weighted with compactness parameter $\alpha \in [0, 1]$ leading to the following dissimilarity term:

$$\psi(p, S) = (1 - \alpha) \cdot d_{rgb}(p, S) + \alpha \cdot d_{xy}(p, S). \tag{3.8}$$

The color space distance d_{rgb} is given by the Euclidean distance in RGB space with

$$d_{rgb}(p, S) = \sqrt{(p_r - S_r)^2 + (p_g - S_g)^2 + (p_b - S_b)^2} \tag{3.9}$$

where $p_{r/g/b}$ and $S_{r/g/b}$ are the values of the red, green, and blue color channels. The color of a superpixel S is the mean color of all pixels assigned to it. As boundary evolution is sensitive to strong image gradients, this effect is reduced by low-pass filtering the images.



(a) Initialization (b) One Interation (c) Five Iterations (d) Convergence

Figure 3.11: Iterative superpixel boundary evolution. The images show segmentation results with an increasing number of iterations from left to right. The segmentation was initialized with compactness parameter $\alpha = 0.98$ and 486 superpixels.

The Euclidean distance d_{xy} between pixel coordinates $p_{x/y}$ and superpixel center $S_{x/y}$ is normalized by the initial superpixel size r:

$$d_{xy}(p, S) = \frac{\sqrt{(p_x - S_x)^2 + (p_y - S_y)^2}}{r}. \tag{3.10}$$

Equation 3.8 shows the function of compactness parameter α. For $\alpha = 0$, the boundary evolution is solely based on color information and completely ignores spatial relationships, leading to very irregular shapes. In contrast, with $\alpha = 1$, the segmentation uses only the spatial distance while ignoring appearance information. This leads to very regular shapes, but without any correlation to the image content. Figure 3.10 gives qualitative examples of segmentations with varying degrees of α.

Local Design

One key aspect of the presented algorithm is its strongly local design. Computing the similarity measure requires only information about the boundary pixel p and its immediate superpixel neighbors. This information is accessible within the 3×3 area, or 8-neighborhood, around pixel p. The segmentation algorithm can modify the boundaries on this local scale and does not require any information about the global segmentation as long as superpixels remain connected. To guarantee for the global segmentation that superpixels are not split,

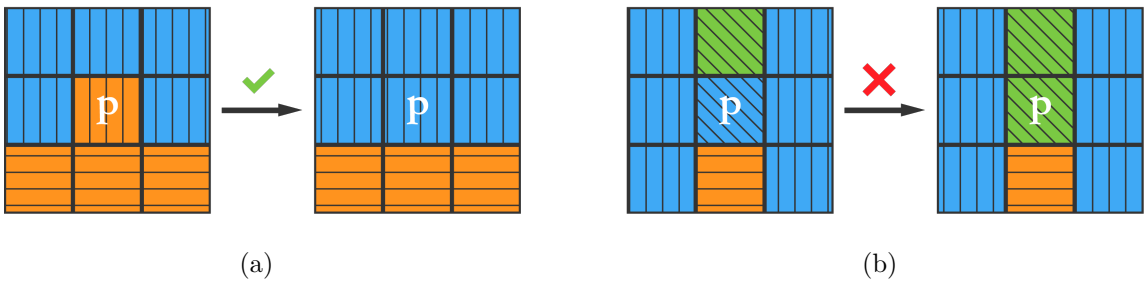


Figure 3.12: Local boundary evolution during superpixel segmentation. The 3×3 grid shows the local 8-neighborhood around pixel p. The superpixels are represented by different colors. The shading represents the similarity between pixels. In (a), pixel p is more similar to the upper superpixel than to its current one and, therefore, assigned to it. The assignment is valid because all superpixels remain connected. In (b), the similarity of pixel p to three superpixels is computed. However, even though p would fit better to the upper superpixel, it cannot be assigned to it because this would split the blue superpixel.

it is sufficient to ensure that they are not split on the local level. This means that connected pixels that belong to the same superpixel in the 3×3 window must remain connected if pixel p is reassigned. Figure 3.12 shows examples of a valid and an invalid reassignment.

The local design has several advantages. First, it reduces computations because the similarity term does not have to be computed for all pixels, but only for the subset of boundary pixels. Second, by checking connectivity on the local level, the global connectivity of superpixels is maintained. Third, it allows a parallel implementation on the GPU because the local operations are independent of each other. In this way, every boundary pixel is processed simultaneously to increase execution speed.

Algorithm 2 shows pseudo-code for an implementation of this superpixel segmentation. In case the algorithm is executed in parallel, there are two necessary implementation requirements. First, the processes modifying the boundary pixels must be started in non-overlapping windows to avoid that two neighboring pixels are modified at exactly the same time. Second, the mean color and position of superpixels have to be updated during segmentation with atomic operators to avoid that updates are overwritten.

Complexity Analysis

The presented superpixel algorithm is very efficient in terms of time and space as the complexity analysis will now show.

Algorithm 2 Superpixel Segmentation

```
Input: image 𝓘
Output: superpixel segmentation 𝓢
Define n_4(p): 4-neighborhood around p
Define m(p): supervoxel that p belongs to
Define c(p): true, if connectivity around p maintained

Initialize superpixel segmentation 𝓢
while not converged do
  for all Pixel p ∈ 𝓘 do
    if p is boundary pixel then
      d ← ψ(p, m(p))
      for all S ∈ 𝓢: ∃q ∈ n_4(p) ∧ m(q) = S do
        if ψ(p, S) < d then
          if c(p) then
            m(p) ← S
            d ← ψ(p, S)
          end if
        end if
      end for
    end if
  end for
end while
```

In the worst case, the superpixel algorithm computes the similarity measure for all pixels in each iteration. Let $|\mathcal{I}|$ be the size of an image an let K be the number of iterations. The time complexity then is in $\mathcal{O}(|\mathcal{I}| \cdot K)$. Given a constant number of iterations, the algorithm is linear in the number of pixels.

Regarding space complexity, the algorithm requires storing the image and the set of superpixels $\mathcal{S}$. This leads to a space complexity in $\mathcal{O}(|\mathcal{I}| + |\mathcal{S}|)$. Because the number of superpixels is lower or equal to the number of pixels, the space complexity is linear in the number of pixels.

For superpixel segmentation in general, every pixel needs to be processed. Therefore, a linear time complexity is optimal. Regarding space complexity, every pixel must be assigned to a superpixel. Therefore, a linear space complexity is optimal. There are more efficient representations, *e.g.*, by only storing boundary pixels, but this does not change the complexity class.

3.2.3 Evaluation

The evaluation compares the superpixel algorithm from Section 3.2.2 to six segmentation algorithms by using the new compactness metric in addition to the established metrics.

Experimental Setup

The evaluation used the full Berkeley segmentation dataset (BSDS) [102]. It contains 300 images with a total of $1,633$ human ground truth annotations. The images cover a broad variety of scenes including humans, animals, landscapes, and buildings. The image resolutions are 481×321 pixels and 321×481 pixels.

The superpixel segmentation algorithms used in this evaluation are the normalized cuts implementation (NC) from [114], SLIC [19], TurboPixels (TP) [95], Superpixel Lattices (LATTICE) [112], and entropy rate superpixels (ERS) [97]. Parameters were set according to the values recommended by the authors. The evaluation of SLIC also included the maximum range of the weighting parameter by setting it to $m = 0$ and $m = MAXINT$. The segmentation algorithm proposed by Felzenszwalb and Huttenlocher (FH) [63] served as an additional benchmark algorithm. It does not aim to compute superpixels, but it achieves very accurate image segmentations and is commonly used in superpixel evaluations. By adjusting the parameters of FH, the number of image segments, *i.e.*, superpixels, can be controlled.

The evaluation compares the segmentation performance with four metrics. Besides the compactness metric (CO) introduced in Section 3.2.1, the following metrics were used.

Boundary recall (BR) measures the fraction of ground truth boundaries overlapping with superpixel boundaries. It is well suited for superpixel segmentations because it does not penalize the redundant boundaries resulting from oversegmentation. Consequently, the boundary recall metric is the most often used metric in the context of superpixels. When computing boundary recall, some authors allow a certain tolerance for the boundary overlap. This evaluation does not use such a tolerance.

The beneficial property of not penalizing redundant boundaries is at the same time the main drawback of this metric. The boundary recall tends to improve with an increase of boundary pixels. It achieves a perfect score of 1 if the whole image consists of superpixel boundaries. Clearly, this would not be a good segmentation. Therefore, it is important to use another metric that penalizes excess boundaries and that acts as a balancing factor. The discussion will show that the compactness metric is well-suited for this task.

A pixel p is called a boundary pixel if one of its 4-neighbors belongs to another segment. Let $b_G(p)$ and $b_{\mathcal{S}}(p)$ be true if pixel p of image $\mathcal{I}$ is a boundary pixel in the ground truth segmentation G or the superpixel segmentation $\mathcal{S}$ and let $|\cdot|$ give the number of pixels. The boundary recall $BR(\mathcal{S}, G)$ then is

$$BR(\mathcal{S}, G) = \frac{|\{p \in \mathcal{I} : b_{\mathcal{S}}(p) \wedge b_G(p)\}|}{|\{p \in \mathcal{I} : b_G(p)\}|}. \tag{3.11}$$

Undersegmentation error (UE) measures the "bleeding" [95] caused by undersegmentation. Superpixels are meant to oversegment an image, *i.e.*, each ground truth object should be partitioned by a set of superpixels and each superpixel should belong to only one ground truth segment. If superpixels overlap with more than one ground truth segment, undersegmentation occurred. This metric measures the overlap of superpixels with more than one ground truth segment and was introduced by Levinshtein *et al.* [95]. It is often used for superpixel evaluation, *e.g.*, by [19, 97, 163, 180, 183].

Let g_i be a ground truth segment of ground truth segmentation G and $|\cdot|$ give the number of pixels. Then, following [95], the undersegmentation error of superpixel segmentation $\mathcal{S}$ with respect to one ground truth segment g_i is

$$UE(\mathcal{S}, g_i) = \frac{\left(\sum_{S \in \mathcal{S} | S \cap g_i \neq \emptyset} |S|\right) - |g_i|}{|g_i|}. \tag{3.12}$$

The undersegmentation error for the whole superpixel segmentation is then

$$UE(\mathcal{S}, G) = \sum_{g_i \in G} \left(UE(\mathcal{S}, g_i) \cdot \frac{|g_i|}{|G|} \right). \tag{3.13}$$

Note that, in contrast to the other metrics, the undersegmentation error can become greater than 1.

Achievable accuracy (AA) measures the accuracy of the best possible segmentation of ground truth objects when using the given superpixels as building blocks. Therefore, it measures one of the core functions of superpixels: their utility to act as atomic primitives. This metric was introduced by Nowozin *et al.* [120] and also used by Liu *et al.* [97].

Let g_i be a ground truth segment of ground truth segmentation G and $|\cdot|$ give the number of pixels. Let $M(\mathcal{S}, g_i)$ be the set of superpixels that has the largest overlap with g_i. Then, following [120] and [97], the achievable segmentation accuracy for one ground truth segment is given by

$$AA(\mathcal{S}, g_i) = \frac{\sum_{S \in M(\mathcal{S}, g_i)} |S \cap g_i|}{|g_i|} \tag{3.14}$$

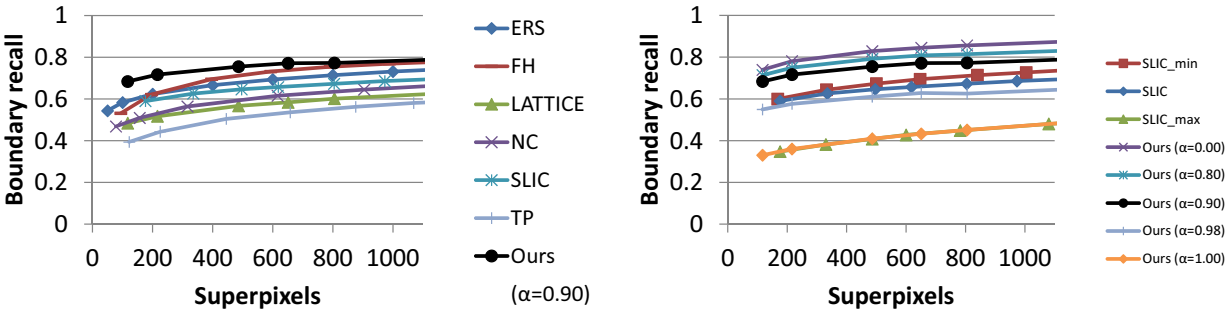
and for all ground truth segments by

$$AA(\mathcal{S}, G) = \sum_{g_i \in G} \left(AA(\mathcal{S}, g_i) \cdot \frac{|g_i|}{|G|} \right). \tag{3.15}$$

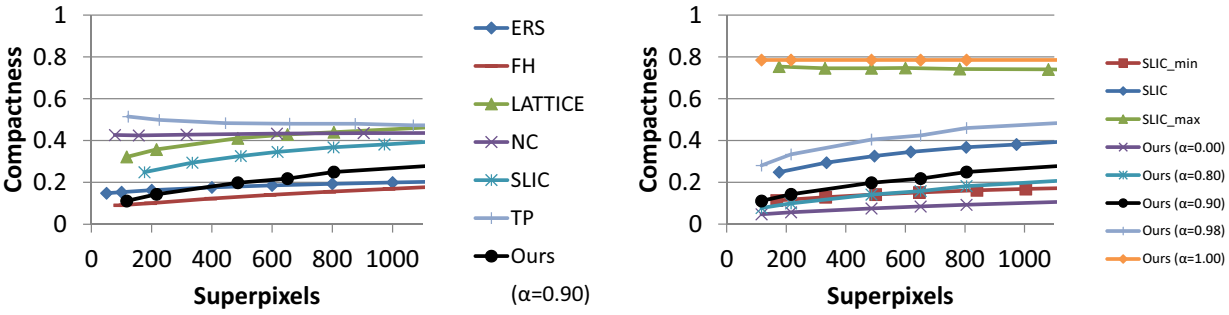
Results and Discussion

In this section, the evaluation results are presented and discussed. A visualization of the results is given in several figures. The graphs in Figure 3.13 show quantitative results for all four metrics. Figure 3.14 shows the correlation between compactness and the other metrics. Figure 3.15 gives example segmentations of all algorithms and Figure 3.16 shows additional qualitative examples of the presented algorithm.

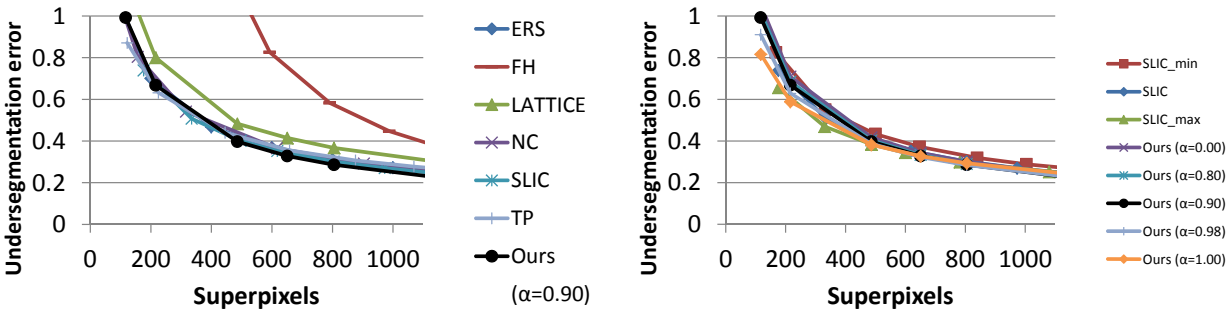
The results in Figure 3.13 show that the performance of the presented algorithm can be effectively controlled by adjusting compactness parameter α. The higher α, the more compact the segmentation becomes. But there is a negative correlation between compactness and the other metrics, in particular boundary recall, as Figure 3.14 shows. The more compact a segmentation, the lower is its boundary recall. This is a strong correlation with



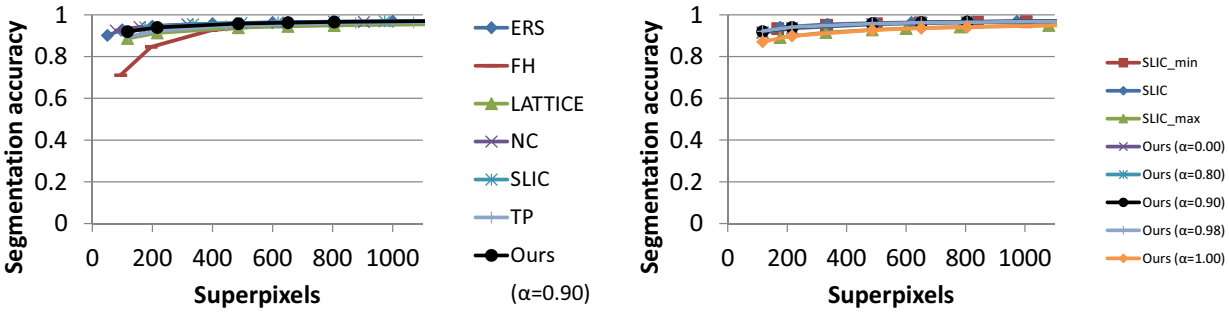
(a) Boundary recall



(b) Compactness

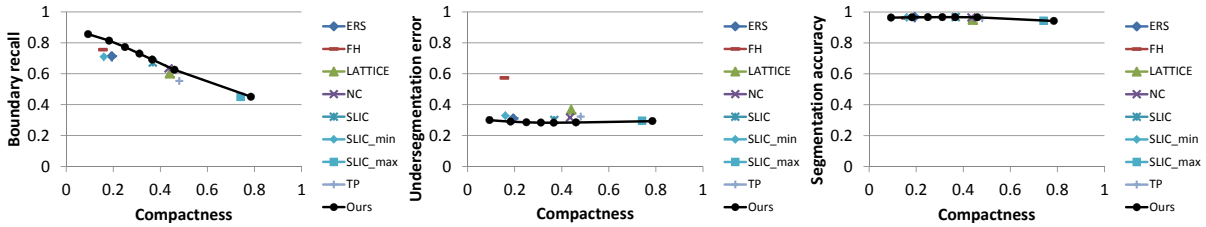


(c) Undersegmentation error



(d) Achievable segmentation accuracy

Figure 3.13: Evaluation of superpixel segmentations with the four metrics boundary recall, compactness, undersegmentation error, and achievable segmentation accuracy. The left graphs show comparisons to the six benchmark segmentations. The right graphs show variations of parameter α compared to the full parameter range of SLIC [19].



(a) Boundary recall over compactness

(b) Undersegmentation error over compactness

(c) Achievable segmentation accuracy over compactness

Figure 3.14: Correlation between compactness and the other superpixel metrics. There is a strong correlation between compactness and boundary recall with a correlation coefficient of -0.93.

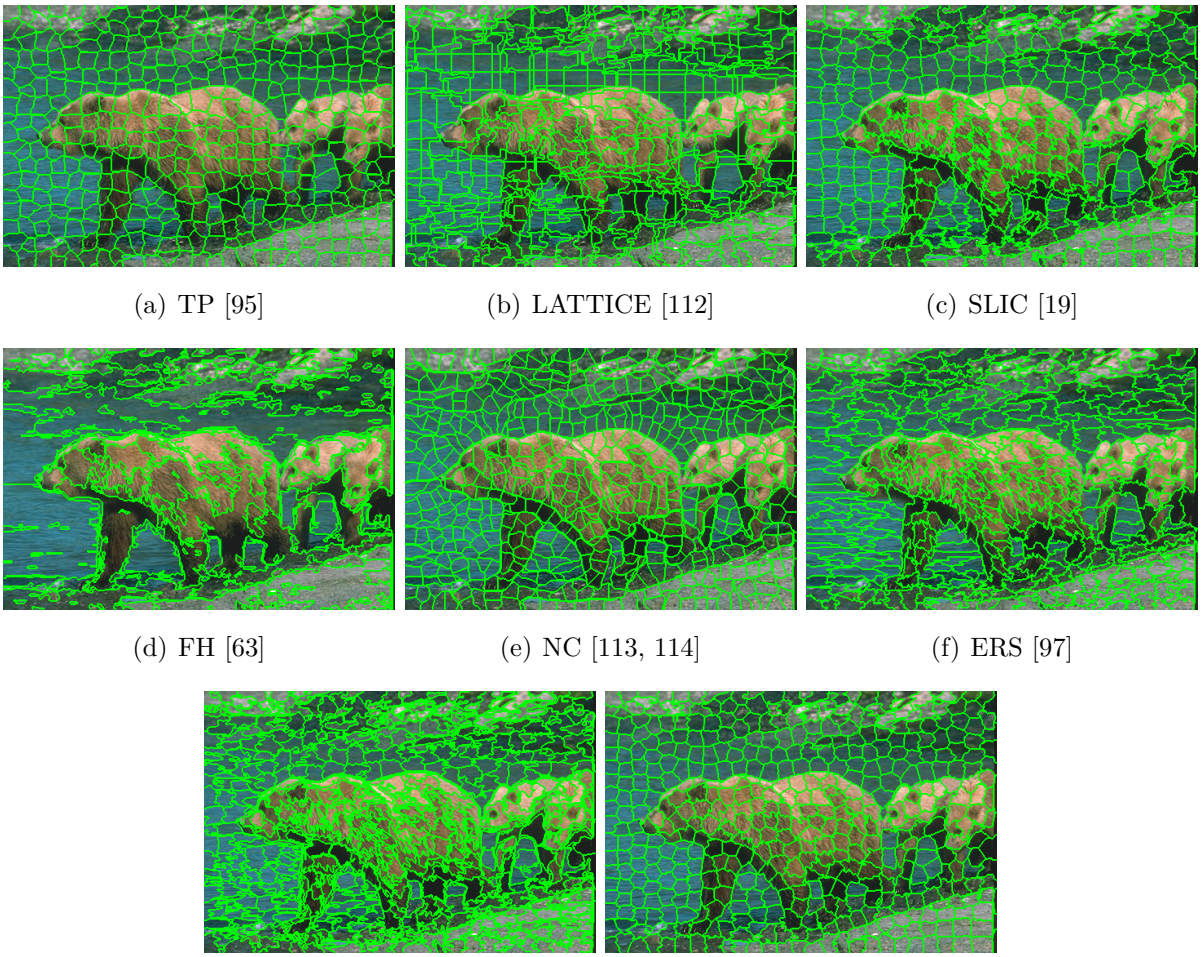
a correlation coefficient of -0.93 and holds for all evaluated segmentation algorithms. It is also in accordance with the definition of compactness: for more accurate segmentations, the superpixels have to adapt better to image boundaries which leads to a decreased compactness, and vice versa. For the other metrics, the correlation coefficient is still negative, but does not support a significant correlation (UE: -0.38, AA: -0.69).

Given the trade-off between compactness and boundary recall, the recommended parameters for the presented algorithm are $\alpha = 0.9$ for accurate and $\alpha = 0.98$ for compact segmentations. With $\alpha = 0.9$, the superpixel algorithm achieves better boundary recall than all other algorithms for all superpixel sizes with similar or better performance for undersegmentation error and achievable segmentation accuracy (Figure 3.13).

A comparison of results in Figure 3.13 with qualitative results in Figure 3.15 shows that segmentations that receive a higher compactness score appear to be visually more compact as well. An additional analysis of the implications of compactness including an example application is given in [3, 4]. In this work, however, the focus regarding superpixels is its extension to supervoxels in 3D.

3.2.4 Conclusion

This section introduced a superpixel segmentation algorithm and showed that it performs equally or better than existing algorithms. In addition, a compactness metric was introduced that measures the compactness of superpixel segmentations. This metric is an additional tool to evaluate superpixel algorithms. In the context of this work, it was shown that there is a significant correlation between boundary recall and compactness.



(a) TP [95] (b) LATTICE [112] (c) SLIC [19]

(d) FH [63] (e) NC [113, 114] (f) ERS [97]

(g) $\text{Ours}_{\alpha=0.90}$ (h) $\text{Ours}_{\alpha=0.98}$

Figure 3.15: Qualitative examples of the evaluated superpixel algorithms. The segmentations consist of 400 to 500 superpixels and show the different qualitative properties of the algorithms.

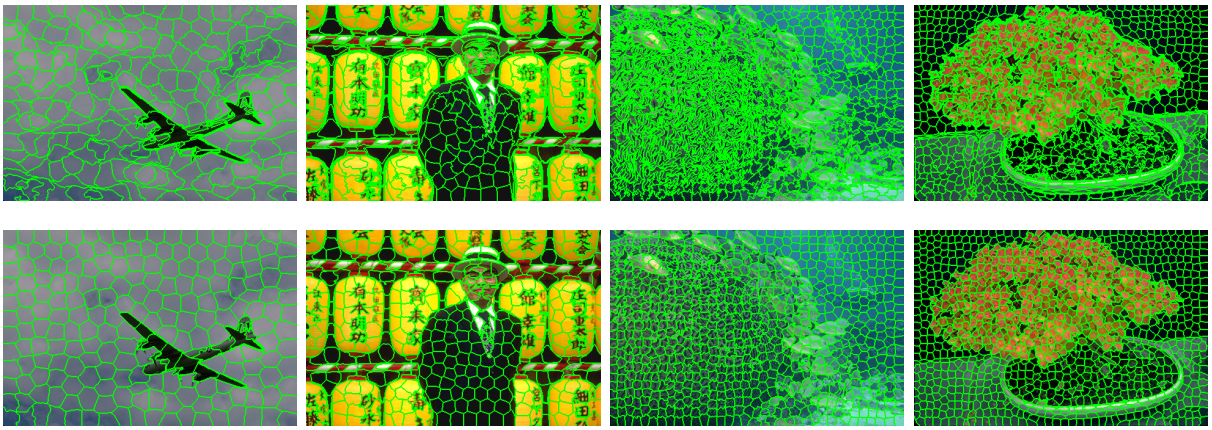


Figure 3.16: Additional qualitative example segmentation with the presented superpixel algorithm. The top row shows results for the recommended compactness parameter $\alpha = 0.9$ and the bottom row for $\alpha = 0.98$. The superpixel numbers from left to right are: 216, 486, 805, and 1107.

In this thesis, the superpixel algorithm serves two purposes. First, it can be used to improve the foreground segmentation for voxel carving with video images as was demonstrated in [2]. Second, it can directly be extended to supervoxels for segmentation of 3D voxel volumes. The next section will show how the properties of the image-based algorithm pass over to volume-based supervoxel segmentation. Additionally, a metric to measure the compactness of 3D supervoxel segmentations will be presented as well.

3.3 Supervoxel Segmentation

This section shows how the concepts of superpixel segmentation and the algorithm introduced in the previous section can be continued in 3D and applied to supervoxel segmentations of voxels. First, a metric is introduced to measure the compactness of supervoxel segmentations. Then, the supervoxel segmentation algorithm is described which is a direct continuation of the superpixel algorithm presented in the previous section. After an evaluation of the algorithm, this section concludes with the introduction of supervoxel graphs that are the basis for the estimation of body parts.

The main contribution presented in this section is an efficient segmentation method for 3D voxel data into supervoxels. By using supervoxels instead of voxels, the number of input elements for the following algorithms is significantly reduced. In the case of pose estimation, for example, the search space will be reduced by several orders of magnitude

as will be shown in Section 4. An additional contribution is a 3D compactness metric for supervoxel segmentations. Further, the supervoxel graph, that is presented in this work, is a special representation that uses the volumetric nature of supervoxels to form connections between them.

3.3.1 Compactness

The concept of compactness, that was introduced for superpixels in Section 3.2, can also be applied to supervoxels. As for superpixels, compactness is a desirable property for supervoxels, too. The reasoning follows the argument for superpixel compactness. Elements of compact supervoxels are close to each other with smooth supervoxel surfaces. Therefore, spatially coherent information is better represented by more compact supervoxels. Also, operations involving the surface voxels of a supervoxel are more efficient because the number of surface voxels and visible voxel faces is smaller for more compact supervoxels with smoother surfaces.

The compactness of a shape in 3D can be measured by setting its volume in relationship to its surface. The more compact, the higher becomes the volume-to-surface ratio of a shape. Bribiesca developed a compactness measure that specifically targets volumes represented by voxels [43, 44] that will be the basis for the supervoxel compactness metric.

Bribiesca measures the compactness of an object composed of voxels based on comparing its inner contact area to the maximum inner contact area of an object that has the same volume [44]. The inner contact area A_c is equal to the number of voxel faces inside the volume and can be computed using the number of voxels and the outer surface area. The surface area A is given by the visible voxel faces on the surface of the object. Let n be the number of voxels and let the surface area of a voxel face be equal to one. Then, the inner contact surface A_c is given by [43]:

$$A_c = \frac{6n - A}{2}. \tag{3.16}$$

Equation 3.16 expresses that the inner contact surface area is given by the total number of voxel faces $6n$ without the visible voxel faces A divided by 2 because each inner contact area is shared by two voxels.

Due to the discretization through voxels, the object with the maximum contact area for a given volume is a cube as was shown in [43]. Then, following Equation 3.16, the maximum

contact area $A_{c_{max}}$ of a cube with side length m consisting of $n = m^3$ voxels and with a surface area of m^2 for each side is given by [43]:

$$A_{c_{max}} = \frac{6m^3 - 6m^2}{2} = 3(m^3 - m^2) = 3(n - n^{\frac{2}{3}}). \tag{3.17}$$

The compactness C of a 3D object consisting of n voxels with visible surface area A is then given by the ratio of its contact surface area to the maximum contact surface area of an object with the same volume [44]:

$$C = \frac{A_c}{A_{c_{max}}} = \frac{3n - \frac{A}{2}}{3(n - n^{\frac{2}{3}})} = \frac{n - \frac{A}{6}}{n - n^{\frac{2}{3}}}. \tag{3.18}$$

Similar to the compactness of superpixels, this compactness measure can now be used to formulate a compactness metric for supervoxels. Let $\mathcal{V}$ be a set of supervoxels and let C_V be the compactness of supervoxel V as given by Equation 3.18 with $|\cdot|$ giving the number of voxels. Then, the compactness $C_\mathcal{V}$ of a supervoxel segmentation $\mathcal{V}$ is

$$C_\mathcal{V} = \sum_{V \in \mathcal{V}} \left(C_V \cdot \frac{|V|}{|\mathcal{V}|} \right). \tag{3.19}$$

This metric is easy to calculate because it only involves counting the number of voxels and visible surface areas. By normalizing with the size of the supervoxels relative to the sum of all voxels, the metric is robust to variations of the sizes of supervoxels.

3.3.2 Supervoxel Segmentation of Volumes

The supervoxel segmentation for voxel volumes is the continuation of the superpixel segmentation for images that was introduced in Section 3.2. Instead of working on 2D pixels of an image lattice, it segments 3D voxels arranged in a regular grid. The basic principles remain the same and the properties of the superpixel segmentation are carried over to the domain of 3D segmentation.

Algorithm

The supervoxel algorithm initializes the segmentation with a regular grid of cuboids. The size of the cuboids is given by the initial supervoxel size. The grid partitions the voxel volume into equally sized cells and each voxel is assigned to its corresponding supervoxel. Then, after initialization, the supervoxel segmentation iteratively evolves the surfaces of the initial segmentation towards convergence. In each iteration, the similarity of each surface voxel to its immediate six neighboring supervoxels is computed. If it is more similar to a neighboring supervoxel, it is assigned to it, thereby improving the overall segmentation. This process is repeated for a given number of iterations or until convergence.

The boundary evolution is guided by the dissimilarity measure $\psi(v, V)$ between voxel v and supervoxel V. In each iteration, it is applied to all supervoxels in the 6-neighborhood around voxel v. It measures the dissimilarity based on spatial proximity d_{xyz} and difference between normals d_n weighted with compactness parameter $\alpha \in [0, 1]$:

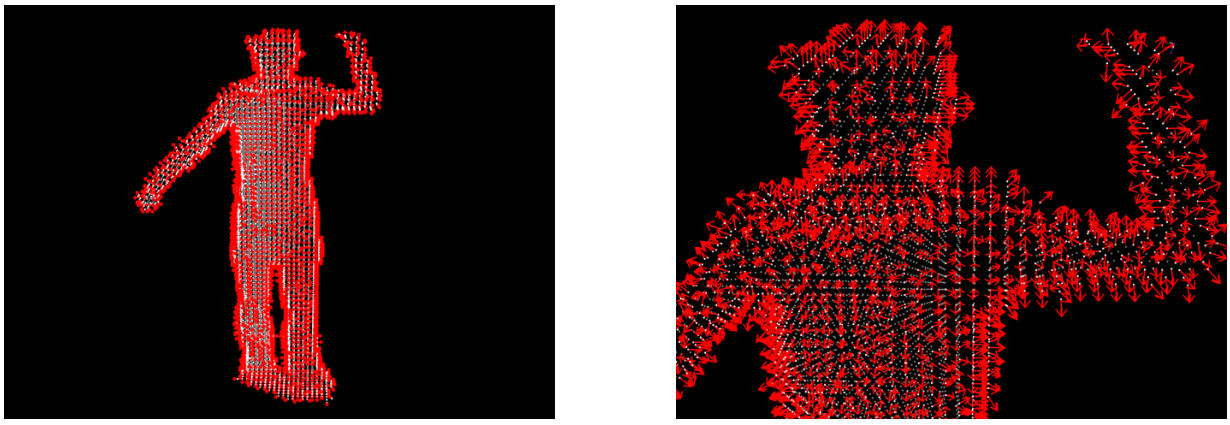
$$\psi(v, V) = (1 - \alpha) \cdot d_n(v, V) + \alpha \cdot d_{xyz}(v, V). \tag{3.20}$$

The compactness parameter α serves the same purpose as for superpixels. It weights the influence of spatial proximity on surface evolution. With $\alpha = 0$, only the surface voxels are modified and the supervoxel surfaces can stretch out leading to less compact supervoxels. With $\alpha = 1$, the supervoxel segmentation is identical to k-means clustering and leads to very compact supervoxels. However, in contrast to image segmentation, where every pixel is occupied and $\alpha = 1$ leads to meaningless segmentations, this is not the case for supervoxels. Here, the presence of a voxel carries meaning in itself because it describes the existence of an object at this position. Therefore, $\alpha = 1$ is indeed a viable option for supervoxel segmentations.

The spatial similarity d_{xyz} is given by the Euclidean distance between voxel v and supervoxel V with center coordinates $v_{x/y/y}$ and $V_{x/y/y}$ normalized by the initial supervoxel size r:

$$d_{xyz}(v, V) = \frac{\sqrt{(v_x - V_x)^2 + (v_y - V_y)^2 + (v_z - V_z)^2}}{r}. \tag{3.21}$$

The similarity d_n is based on normals. It compares the normals of surface voxels with the mean supervoxel normal in Euclidean space. Only surface voxels contribute towards this measure as inner voxels have no normals. A surface voxel is a voxel with at most



(a) Full view (b) Zoomed view

Figure 3.17: Voxel normals. The images show the normals of the surface voxels originating at the voxel centers.

five neighboring voxels, *i.e.*, at least one of the six sides must be visible. Examples of voxel normals are shown in Figure 3.17. The mean supervoxel normal is the sum of all surface voxel normals normalized by the number of surface voxels. The length of the mean supervoxel normal does not necessarily have to be 1. In contrast to the color distance used for superpixels, it is measured in the same space as the spatial similarity and comparison to the spatial distance is more meaningful. With voxel normal $v_{nx/ny/nz}$ and mean supervoxel normal $V_{nx/ny/nz}$, the similarity d_n is given by

$$d_n(v, V) = \sqrt{(v_{nx} - V_{nx})^2 + (v_{ny} - V_{ny})^2 + (v_{nz} - V_{nz})^2}. \tag{3.22}$$

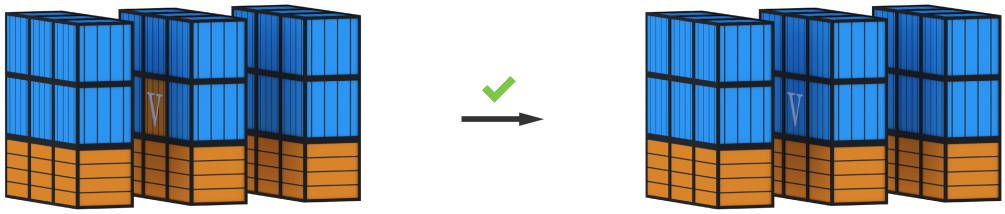
In contrast to the superpixel algorithm, this algorithm does not include color information. Most of the voxels are inside the volume and, therefore, colorless. Also, the surface voxels are generally not colored during voxel carving. However, it would be possible to use color information only for surface voxels instead of normals in Equation 3.22. However, as discussed in Section 2.1.3, color representations can differ between cameras leading to non-reliable information and are not considered in the context of this work.

Similar to the superpixel segmentation, the supervoxel segmentation only works on the surface of supervoxels. This reduces the number of computations required in each iteration substantially. Further, it guarantees that connected supervoxel parts remain connected during segmentation and cannot be ripped apart. Algorithm 3 shows a pseudo-code implementation of the supervoxel segmentation.

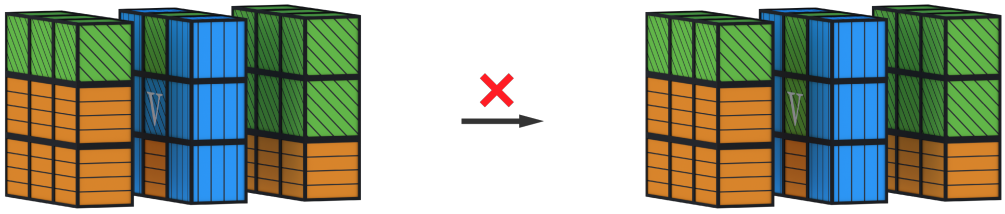
Algorithm 3 Supervoxel Segmentation

Input: voxel grid $\mathfrak{V}$
Output: supervoxel segmentation $\mathcal{V}$
Define $n_6(v)$: 6-neighborhood around v
Define $m(v)$: supervoxel that v belongs to
Define $c(v)$: true, if connectivity around v maintained

Initialize supervoxel segmentation $\mathcal{V}$
while not converged **do**
  for all Voxel $v \in \mathfrak{V}$ **do**
    if v is boundary voxel **then**
      $d \leftarrow \psi(v, m(v))$
      for all $V \in \mathcal{V}$: $\exists q \in n_6(v) \wedge m(q) = V$ **do**
        if $\psi(v, V) < d$ **then**
          if $c(v)$ **then**
            $m(v) \leftarrow V$
            $d \leftarrow \psi(v, V)$
          end if
        end if
      end for
    end if
  end for
end while



(a) Valid change



(b) Invalid change

Figure 3.18: Boundary evolution for supervoxel segmentation. The similarity between boundary voxel v and the supervoxels in its 6-neighborhood is computed. Then, v is assigned to the most similar supervoxel. Colors represent supervoxels and the voxel shading the similarity between voxels. In (a), voxel v fits better to the blue supervoxel and is assigned to it. This is a valid reassignment because both supervoxels remain connected in the 26-neighborhood around v. In (b), even though voxel v would fit better to the green supervoxel, it cannot be reassigned because this would locally split the blue supervoxel.

Local design

Like the superpixel segmentation, the algorithm strictly follows a local design and only works on boundary voxels. The advantages are the same as for the superpixel segmentation: reduction of similarity measurements because only a subset of voxels is evaluated in each iteration, connected supervoxels remain connected, and a parallel implementation on the GPU is possible because neighborhoods can be evaluated independently of each other.

To guarantee that supervoxels remain connected, it is sufficient to enforce connectivity in the local neighborhood around boundary voxel v. If supervoxels remain connected in the local neighborhood, they also remain connected in the global segmentation. Connectivity means that all voxels in the 26-neighborhood around voxel v, which belong to the same supervoxel, must be connected even if v is reassigned. Figure 3.18 shows two examples of valid and invalid reassignments.

3.3.3 Supervoxel Graphs

Similar to superpixels, there is no predetermined structure for supervoxel segmentations. In particular, given a particular supervoxel, its neighborhood is not deterministic. Neighborhoods, however, are often important for algorithms because they introduce structure into the segmentation. This structure can then be used by algorithms to, for example, iterate over the segmentation.

The relationships between supervoxels can be represented by adjacency graphs that connect neighboring supervoxels [121]. An adjacency graph is well-suited to structure supervoxel segmentations, but it is limited to the immediate neighborhood around supervoxels. In this work, the connectivity between two supervoxels follows a different criterion that is based on the volumetric nature of supervoxel segmentations. This leads to a different representation: the supervoxel graph.

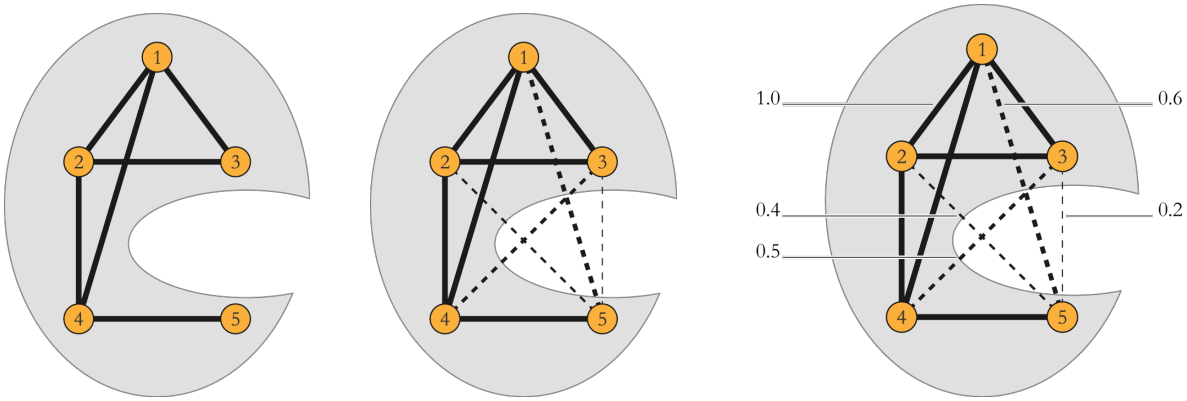
The supervoxel graph is not limited to the neighborhoods around supervoxels, but it connects supervoxel centers across arbitrary distances as long as the connection remains completely inside the segmented volume. This is important because the connections of the supervoxel graph will later be used as candidates for rigid body parts in the pose estimation algorithm described in Section 4.

Let the vertices of the supervoxel graph be represented by supervoxels $V \in \mathcal{V}$ with edges $e_{ij} \in E$ that are the connections between centers of supervoxels V_i and V_j. Let $c(V_i, V_j)$ be the set of voxels v that lie on edge e in voxel grid $\mathfrak{V}$, both carved ($v = 0$) and existing ($v = 1$) ones. Then, the existence of an edge between two supervoxels is given by

$$e_{ij} \in E \iff \forall v \in c(V_i, V_j) : v = 1. \tag{3.23}$$

Equation 3.23 is rather strict in that it requires that the whole connection between two supervoxels is fully within the carved volume. For real data, however, there are often small defects in the visual hull, especially for thinner limbs like the arms. Following this strict formulation, many edges would be removed only because one single voxel is not occupied. Consequently, these removed edges would also not be potential candidates for body parts.

The probabilistic, or weighted, supervoxel graph relaxes the strict formulation in Equation 3.23 by weighting the connection. The weight is equivalent to the fraction of the



(a) Strict supervoxel graph (b) Weighted supervoxel graph (c) Weighted supervoxel graph with annotations

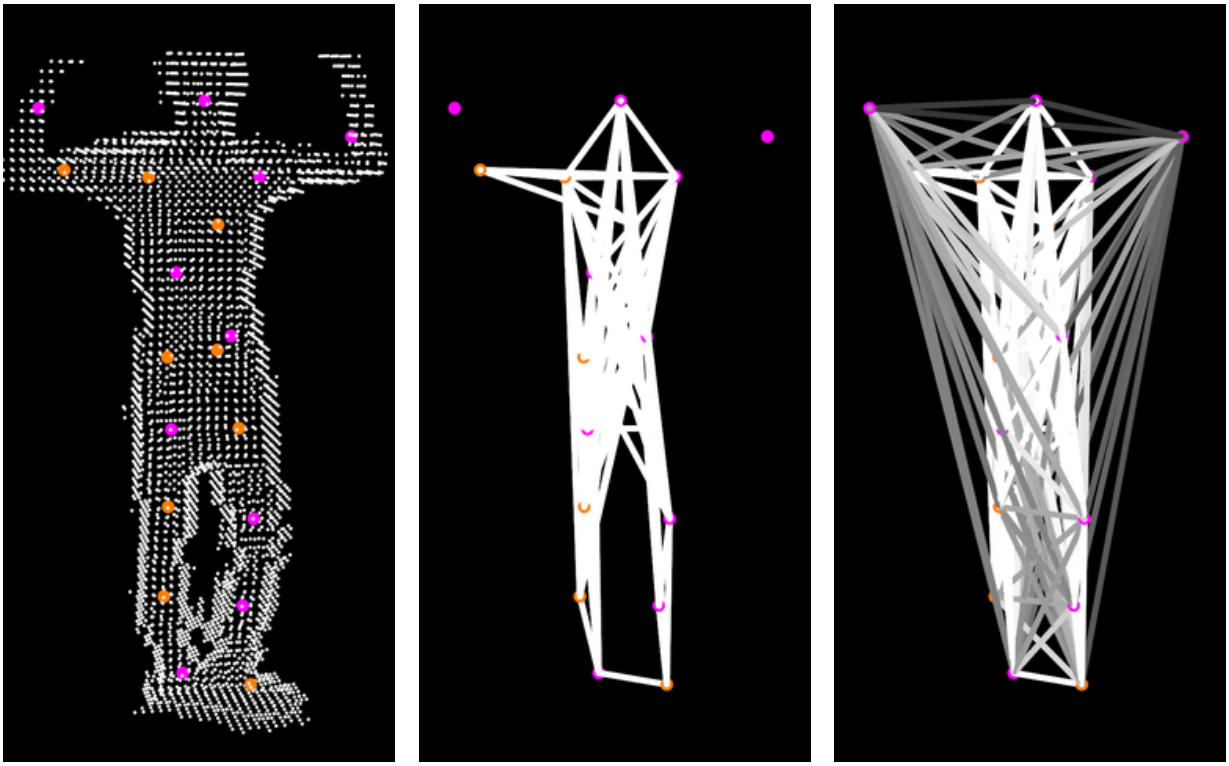
Figure 3.19: Supervoxel and weighted supervoxel graph. The circles represent supervoxel centers inside a volume (gray) and the black lines connections between supervoxels. (a) shows only connections between supervoxels if the connection is fully inside the gray volume (*e.g.*, between 3 and 2, but not between 3 and 5). The thickness of the dashed lines in (b) represent the weight of connections for the fully connected weighted supervoxel graph. (c) is similar to (b), but additionally shows example numbers for the connection weights.

connection that is within the volume. Let $|\cdot|$ give the number of voxels. Then, the weight $w(e_{ij})$ of an edge is given by

$$w(e_{ij}) = \frac{|\bigcup v \in c(V_i, V_j) : v = 1|}{|\bigcup v \in c(V_i, V_j)|}. \tag{3.24}$$

Figure 3.19 shows a visual explanation for both versions of the supervoxel graph. Figure 3.20 shows a voxel segmentation together with supervoxel centers and the resulting strict and weighted supervoxel graphs for an example from the UMPM dataset [162].

Another potential representation form that will not be further discussed in the context of this work are supervoxel grids. Similar to superpixel lattices for images, a supervoxel grid maintains the regular structure analogous to voxel grids. In particular, each supervoxel can be assigned an unique identifier that determines its position in the grid and its neighbors. This representation can more easily be traversed than a general graph. Similar to our work about superpixel lattices in [4], it would be possible to extend the supervoxel segmentation to guarantee a grid structure; however, this is not required here.



(a) Voxels with supervoxel centers (b) Strict supervoxel graph (c) Weighted supervoxel graph

Figure 3.20: Supervoxel graph examples. (a) shows the white voxel centers together with colored supervoxel centers. (b) shows the strict supervoxel graph. The outer supervoxels are not connected to any other supervoxel because the connection would partially be outside the bounding volume. (c) shows a fully connected weighted supervoxel graph with lighter colors indicating a higher weight.

Complexity Analysis

This section presents the complexity analysis of the supervoxel segmentation and for the supervoxel graph algorithms, both for time and space complexity.

Supervoxel Segmentation Let $|\mathfrak{V}|$ be the total number of voxels in the voxel grid and let K be the number of iterations. In the worst case, the similarity of every voxel to all neighboring supervoxels has to be computed in each iteration. The similarity computation involves at maximum six neighbors and is constant. It then follows that the time complexity of the supervoxel segmentation is in $\mathcal{O}(|\mathfrak{V}| \cdot K)$. With a constant number of iterations, the algorithm is linear in the number of voxels.

For real-world scenarios, the actual complexity is much lower. This is because the computation of the similarity term, which is the most expensive part in the algorithm, must only be done for boundary voxels. In general, the number of boundary voxels is significantly smaller than the total number of voxels.

Regarding memory requirements, the algorithm requires storing both the voxel grid $\mathfrak{V}$ and the supervoxels $\mathcal{V}$. This leads to a space complexity in $\mathcal{O}(|\mathfrak{V}| + |\mathcal{V}|)$.

Supervoxel Graphs The time and space complexity for the supervoxel graph is quadratic in the number of supervoxels $|\mathcal{V}|$. In the worst case, there is a connection between every pair of supervoxels that must be computed and stored. Therefore, the time and space complexities are both in $\mathcal{O}(|\mathcal{V}|^2)$. This complexity class applies to the normal as well as the weighted supervoxel graphs.

3.3.4 Evaluation

This section presents the evaluation of the supervoxel algorithm developed during this thesis and the supervoxel algorithm presented by Papon *et al.* [121]. The first part discusses metrics based on their applicability to supervoxels before presenting the evaluation results in the second part.

Metrics

There is a set of well-established metrics for superpixel segmentation, namely boundary recall, undersegmentation error, and achievable segmentation accuracy, that can also be

applied to supervoxel segmentation of volumes as Xu and Corso [182] showed. Even though these metrics target fully occupied volumes created by stacking images, they can also be applied to sparse volumes created by voxel carving as will be discussed in this section. These metrics are complemented by the 3D compactness metric for supervoxels presented in Section 3.3.1.

In contrast to [182], Papon *et al.* [121] followed a different evaluation method. They projected the supervoxels back into the image plane and applied conventional boundary recall and undersegmentation error metrics. This, however, is only well-suited for surfaces as it does not capture information inside of volumes.

3D Boundary Recall (3D BR) The 3D boundary recall measures the overlap of segmentation boundaries with ground truth boundaries. A boundary voxel has at least one neighbor in its 6-neighborhood that belongs to a different supervoxel or ground truth segment. In the sparse voxel grid, surface voxels, *i.e.*, voxels with at least one free voxel face, are also counted as boundary voxels.

While boundary recall is an excellent measure for superpixels, its usefulness for sparse voxel grids is limited. The surface voxels represent a large portion of the ground truth boundaries. They are always correctly classified by definition, thus biasing the metric. In addition, boundaries inside of volumes are not as clearly marked as is the case for images or image stacks. With the absence of color and normal information inside volumes, it is hard to segment the boundaries correctly. For example, it is difficult to decide where the shoulder ends and the arm begins. Given these reasons, the 3D boundary recall should be treated only as an indicator of segmentation accuracy.

Let $b_G(v)$ and $b_{\mathcal{V}}(v)$ be true if voxel $v \in \mathfrak{V}$ is a boundary voxel in the ground truth segmentation G or the supervoxel segmentation $\mathcal{V}$ and let $|\cdot|$ give the number of voxels. Following [182], the 3D boundary recall $BR(\mathcal{V}, G)$ then is

$$BR(\mathcal{V}, G) = \frac{|\{v \in \mathfrak{V} : b_{\mathcal{V}}(v) \wedge b_G(v)\}|}{|\{v \in \mathfrak{V} : b_G(v)\}|}. \tag{3.25}$$

3D Undersegmentation Error (3D UE) The 3D undersegmentation error measures how much supervoxels overlapping with one ground truth segment reach out into other ground truth segments. It faces a similar problem as the 3D boundary recall. Due to non-existent color and normal features inside volumes, it is hard to correctly classify boundaries, thus leading to an increased error. This effect can be quite severe. If only one voxel reaches into another ground truth segment, the whole supervoxel contributes to the error.

Following [182] based on [95], let G be the set of ground truth segments g represented by voxels and let $\mathcal{V}$ be the set of supervoxels V with $|\cdot|$ giving the number of voxels. Then, the 3D undersegmentation error $UE(\mathcal{V}, g_i)$ of ground truth segment g_i is

$$UE(\mathcal{V}, g_i) = \frac{\left(\sum_{V \in \mathcal{V} | V \cap g_i \neq \emptyset} |V|\right) - |g_i|}{|g_i|}. \tag{3.26}$$

The undersegmentation error for the whole segmentation is then

$$UE(\mathcal{V}, G) = \sum_{g_i \in G} \left(UE(\mathcal{V}, g_i) \cdot \frac{|g_i|}{|G|} \right). \tag{3.27}$$

Similar to the 2D undersegmentation error, the 3D undersegmentation error can be greater than 1, as well.

3D Achievable Segmentation Accuracy (3D AA) The 3D achievable segmentation accuracy is probably the best-suited metric for supervoxel segmentations in sparse voxel grids. It measures the basic function of supervoxels without relying on boundaries alone: how accurately objects can be represented with supervoxels as building blocks.

Let $M(\mathcal{V}, g_i)$ be the set of supervoxels that have the largest overlap with ground truth segment g_i and let $|\cdot|$ give their size in voxels. Following [182] based on [120], the $AA(\mathcal{V}, g_i)$ of one ground truth segment is given by

$$AA(\mathcal{V}, g_i) = \frac{\sum_{V \in M(\mathcal{V}, g_i)} |V \cap g_i|}{|g_i|} \tag{3.28}$$

and for all ground truth segments by

$$AA(\mathcal{V}, G) = \sum_{g_i \in G} \left(AA(\mathcal{V}, g_i) \cdot \frac{|g_i|}{|G|} \right). \tag{3.29}$$

3D Compactness (3D CO) The compactness metric described in Section 3.3.1 is specifically designed for supervoxel segmentations of sparse voxel grids and suffers no drawbacks from absent features inside volumes. It accurately measures the shape of supervoxels and describes the regularity of their surface.

Dataset

There are datasets available for supervoxel segmentation of video sequences [182], of image stacks from medical scans [98], and of RGB-D images [150]. However, the data are either not sparse [98, 182] as is the case of voxels computed with voxel carving or they are only annotated in image space [150] which makes it difficult to evaluate 3D compactness. Therefore, the evaluation uses examples based on the ground truth from the Utrecht Multi-Person Motion (UMPM) dataset [162].

The UMPM dataset [162] is well suited for this work because it contains multi-view video sequences of humans together with ground truth labels of human body parts. These body part labels are used to generate ground truth labels for supervoxel evaluation in the following way. First, two voxel representations are acquired, the first one with voxel carving and the second one by synthetically converting body parts into voxels. The synthetic voxels are sampled from cylinders with fixed widths and the body parts as center axes. Then, each voxel is assigned with the label of the ground truth body part that is closest to it. Figure 3.21 shows examples of voxel carving and synthetic volumes with ground truth labels.

Experimental Setup

The evaluation shows results for the sequence *p1_orthosyn_1* of the UMPM dataset [162] that consists of 2480 frames. In addition, the synthetic cube from Section 3.1.4 is used for runtime evaluation.

The evaluation includes both the presented supervoxel segmentation as well as the voxel cloud connectivity segmentation (VCCS) algorithm presented by Papon *et al.* [121] that is implemented in the PCL library [138]. The evaluation includes several initial supervoxel sizes ranging from approximately 10 cm to 50 cm with voxel sizes of 2.5 cm. VCCS uses a similarity measure that combines distances in color space, Euclidean space, and between features based on normals. All three parts can be weighted with a separate parameter: λ for color, μ for spatial distance, and ϵ for normals. Due to the colorless voxels, the color weight was set to $\lambda = 0$. Then, three parameter sets were used. One that only respects spatial distance ($\mu = 1$, $\epsilon = 0$), one that uses only normals ($\mu = 0$, $\epsilon = 1$), and one with equal weights as presented in [121] ($\lambda = 1$, $\mu = 1$). The supervoxel algorithm developed during this work only requires compactness parameter α. The evaluation includes three variations of α that are similar to the variations used for VCCS: $\alpha = 0$ for non-compact segmentations that only use normals, $\alpha = 1$ for very compact segmentations, and $\alpha = 0.9$

as a setting with a good trade-off between accuracy and compactness. In contrast to VCCS, the presented algorithm is an iterative algorithm and the number of iterations was set to 10 for all experiments.

Results and Discussion

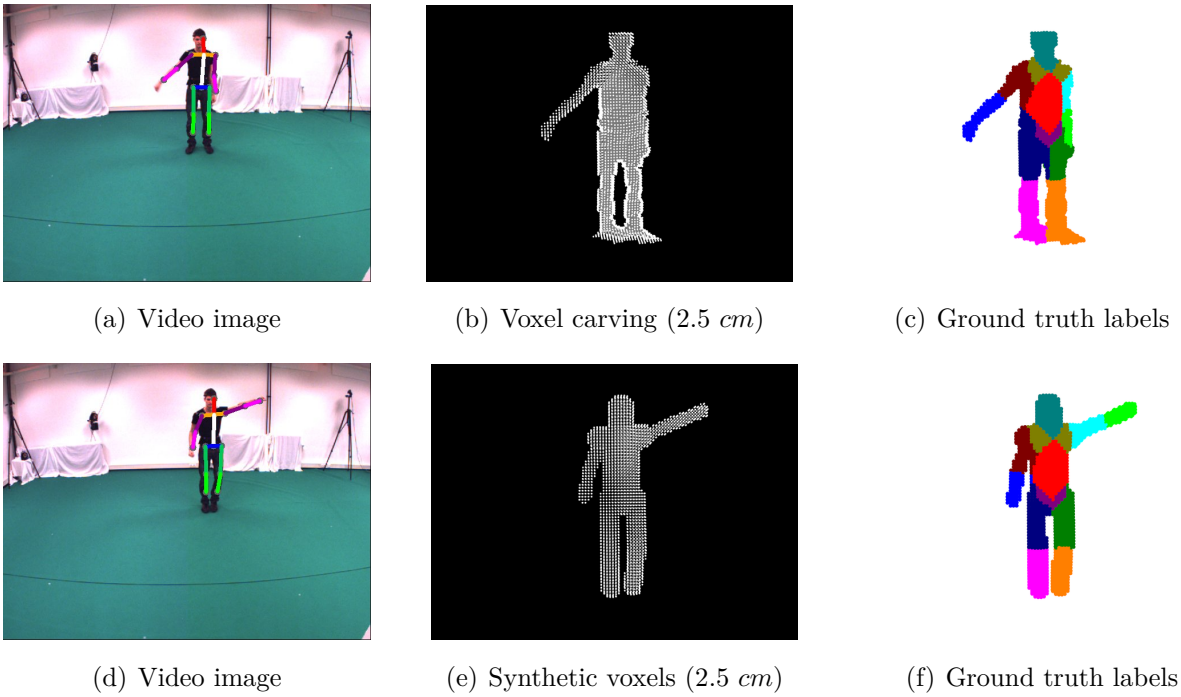
Figure 3.22 shows results for the four metrics 3D boundary recall, 3D compactness, 3D undersegmentation error, and 3D achievable segmentation accuracy. Figure 3.23 shows correlations between compactness and the other three metrics.

Figure 3.22 shows that the performance of the presented algorithm on the four metrics can effectively be controlled with compactness parameter α. The graphs in Figure 3.23 show the correlation between the three metrics and compactness. The correlation coefficient for boundary recall over compactness is negative (correlation coefficient -0.59 for synthetic and -0.68 for real data), but not as significant as for superpixels. However, the correlation analysis shows a significant correlation between undersegmentation error and compactness (synthetic: -0.95, real: -0.95) and segmentation accuracy and compactness (synthetic: 0.89, real: 0.91). While the effect of compactness on boundary recall and undersegmentation error is similar for both superpixel and supervoxel segmentations, it is different for achievable segmentation accuracy. For supervoxels, an increase of compactness also leads to an increase of segmentation accuracy. This positive effect can be attributed to the compactness of body parts in the ground truth. As a consequence, this work will use parameters for a compact segmentation ($\alpha = 1$) because it gives the best results in the context of pose estimation.

In comparison to VCCS, the presented algorithm achieves similar results for comparable settings (for example, VCCS with $\mu = 1$ and $\epsilon = 0$ is similar to $\alpha = 1$). VCCS achieves the best results for boundary recall, whereas the presented algorithm achieves the lowest undersegmentation error and highest segmentation accuracy (with $\alpha = 1$). Both algorithms achieve equally high best values for compactness.

The comparison in Figure 3.22 between synthetic voxels and the result of data from voxel carving shows no apparent differences.

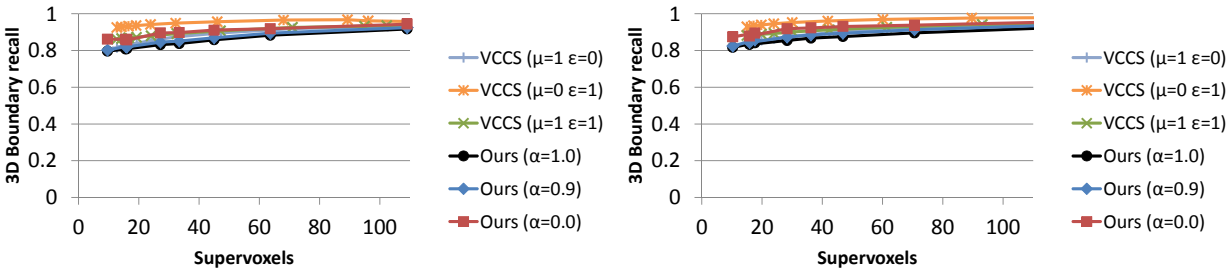
Figure 3.24 shows qualitative results for various numbers of iterations. A qualitative comparison of the various parameter settings for both algorithms is presented in Figure 3.25.



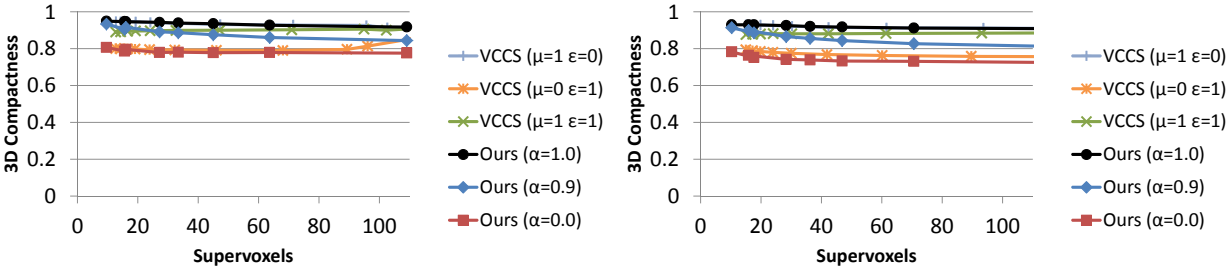
(a) Video image (b) Voxel carving (2.5 cm) (c) Ground truth labels

(d) Video image (e) Synthetic voxels (2.5 cm) (f) Ground truth labels

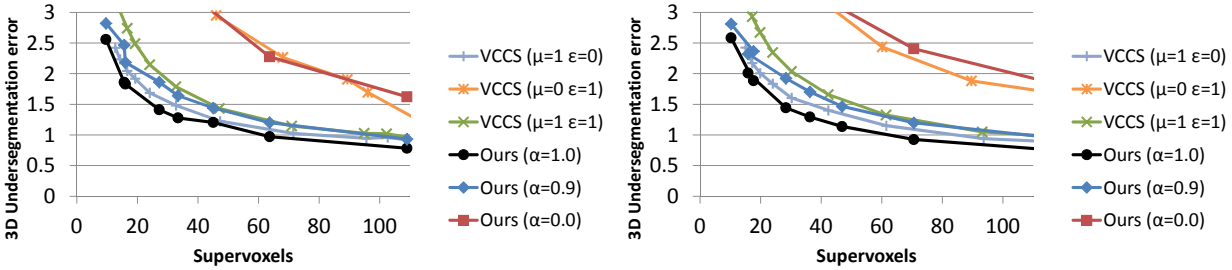
Figure 3.21: Ground truth examples for the evaluation of supervoxel segmentations. The left images show one of the four views of the video sequence from the UMPM dataset [162] with ground truth poses. The middle images show 3D reconstructions with (b) voxel carving and (e) synthetically generated voxels. The right images show voxels colored according to the ground truth label of the body part they are assigned to.



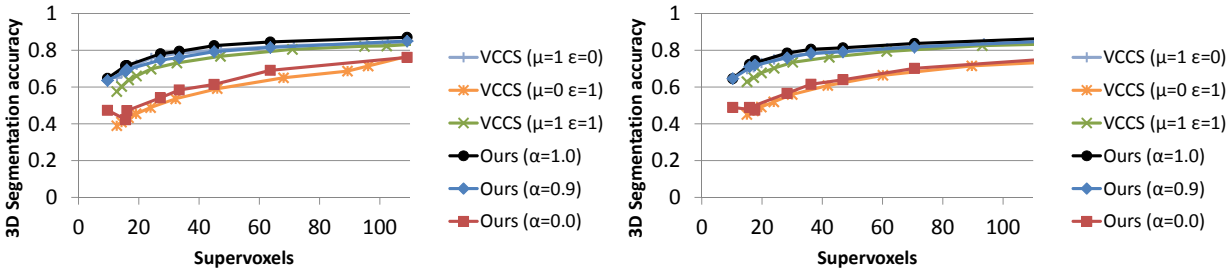
(a) 3D Boundary recall for synthetic (left) and real voxels (right)



(b) 3D Compactness for synthetic (left) and real voxels (right)

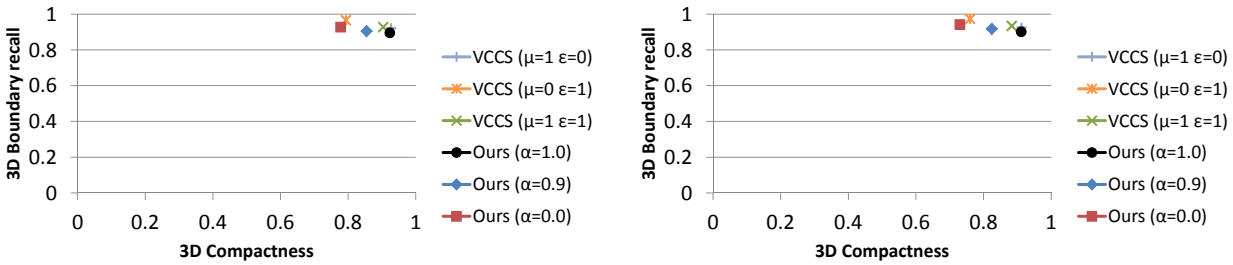


(c) 3D Undersegmentation error for synthetic (left) and real voxels (right)

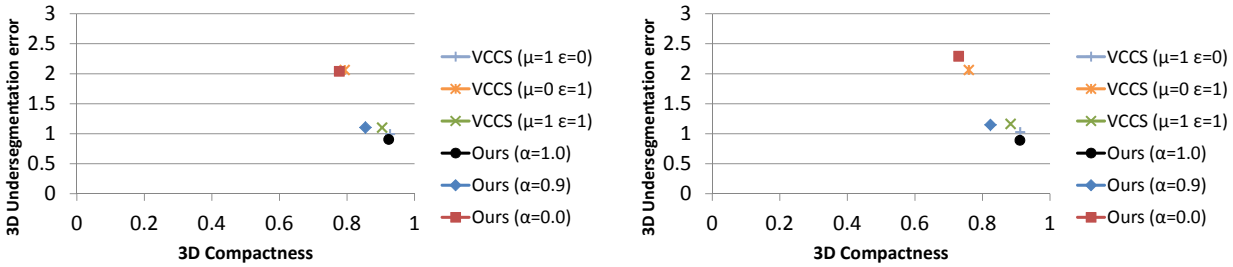


(d) 3D Achievable segmentation accuracy for synthetic (left) and real voxels (right)

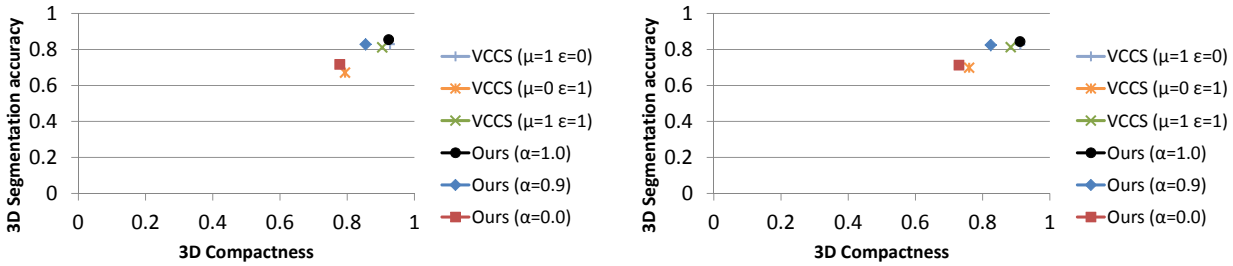
Figure 3.22: Evaluation of supervoxel segmentations with the four metrics 3D boundary recall, 3D compactness, 3D undersegmentation error, and 3D achievable segmentation accuracy. The graphs show comparisons of the proposed algorithm to VCCS [121] using data from the UMPM dataset [162]. The left side shows results with synthetic voxels and the right side with real voxel carving data as the basis for the supervoxel segmentations.



(a) 3D boundary recall over 3D compactness

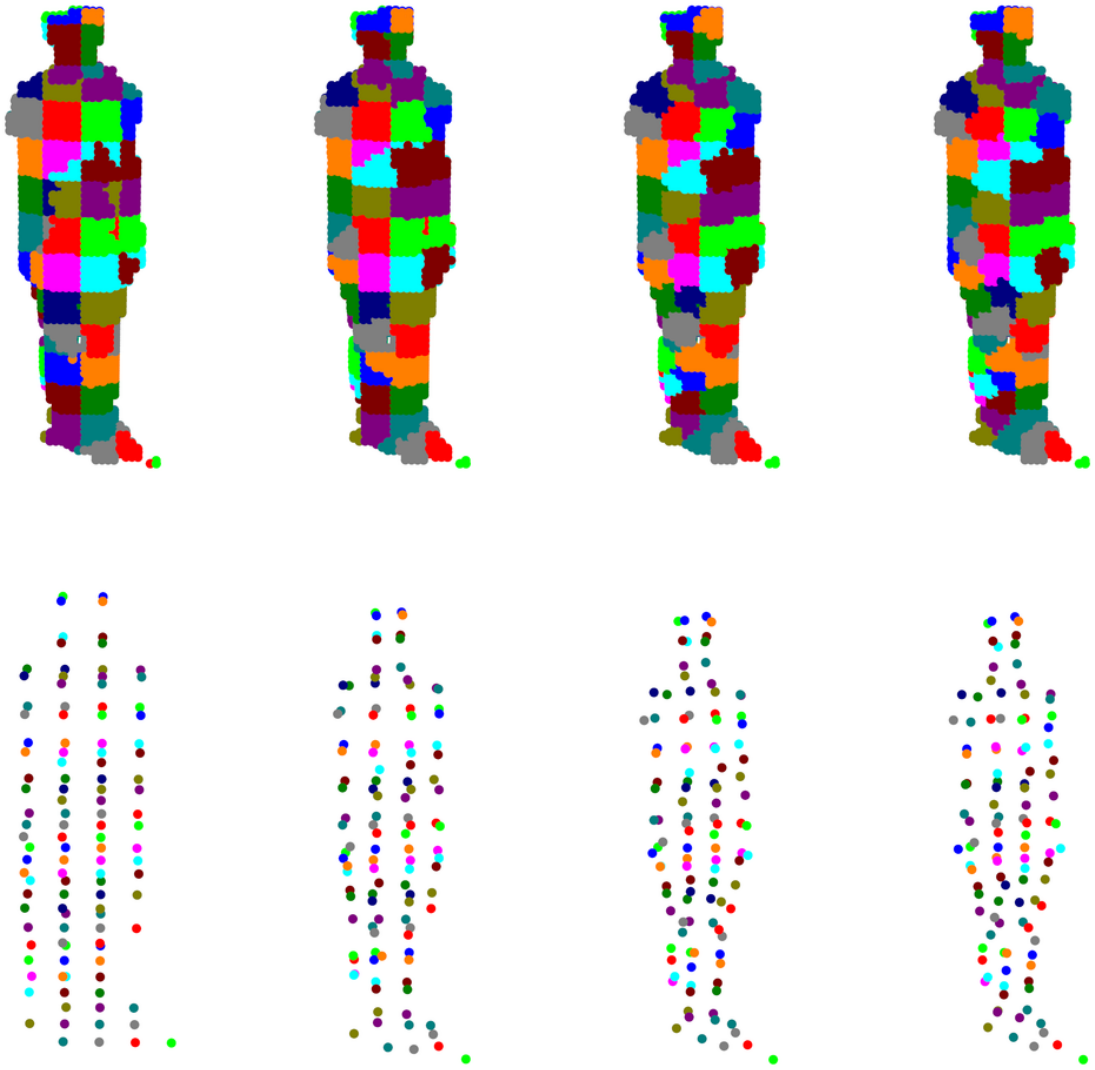


(b) 3D undersegmentation error over 3D compactness



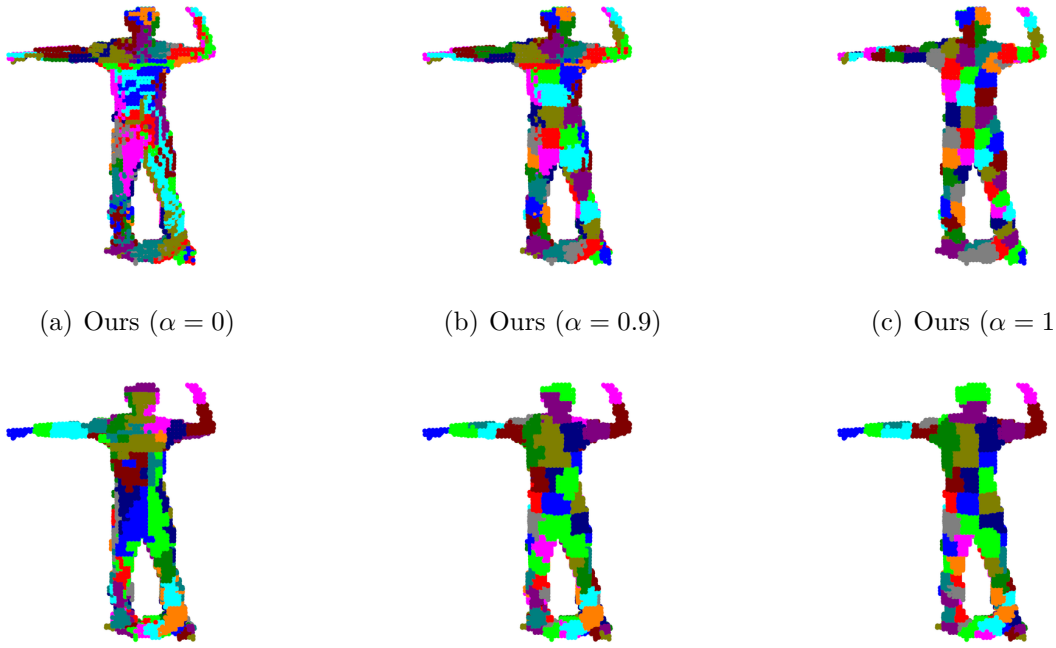
(c) 3D achievable segmentation accuracy over 3D compactness

Figure 3.23: Correlation analysis for supervoxel compactness. The graphs show, from top to bottom, the correlation of boundary recall, undersegmentation error, and achievable segmentation accuracy with compactness. The left column shows results for synthetic voxels and the right column for multi-view voxel carving. There is a strong correlation for undersegmentation error (-0.95 and -0.95) and segmentation accuracy (0.89 and 0.91), for synthetic and real voxel reconstructions, respectively.



(a) Initialization (b) One iteration (c) Five iterations (d) Ten iterations

Figure 3.24: Supervoxel segmentation iterations. The images show from left to right an increasing number of iterations for a supervoxel size of 15 cm with compactness parameter $\alpha = 1$. The top row shows voxels colored according to their supervoxels. The rectangular structure of the initial grid in (a) continuously converges towards more equally sized supervoxels in (d). Note that the partially irregular structure in (a) is the result of not fully filled supervoxel cells. The bottom row shows the centers of supervoxels.



(a) Ours ($\alpha = 0$) (b) Ours ($\alpha = 0.9$) (c) Ours ($\alpha = 1$)

(d) VCCS ($\mu = 0.0$, $\epsilon = 1.0$) (e) VCCS ($\mu = 1.0$, $\epsilon = 1.0$) (f) VCCS ($\mu = 1.0$, $\epsilon = 0.0$)

Figure 3.25: Qualitative comparison of various parameter settings. The top row shows qualitative results of the presented algorithm and the bottom row of VCCS [121] based on data from the UMPM dataset [162]. The voxel size was set to 2.5 cm and the supervoxel size to 15 cm. The segmentations show results for various parameter setting from least compact (left) to most compact (right). The parameter settings in each column correspond to each other, in particular for the least compact (a+d) and most compact (c+f) segmentations.

Runtime Analysis

This section discusses the runtime analysis for both the supervoxel segmentation as well as the strict and weighted supervoxel graph computations.

Both evaluations include the following two scenarios: First, the synthetic cube scenario that was also used for the voxel carving evaluation in Section 3.1.4. This synthetic scenario is used as a benchmark with controlled conditions. Second, the UMPM scenario that shows the first frame of the UMPM dataset sequence $p1\_orthosyn\_1$ with one person. The UMPM scenario is most similar to the main topic of this thesis and will be the basis to determine the supervoxel sizes, and therefore their number, for pose estimation in Section 4. The supervoxel segmentation evaluation includes two additional scenarios: the best and worst case scenarios from Section 3.1.4 that use empty and full voxel grids.

The evaluations included four supervoxel sizes: 10 cm, 15 cm, 20 cm, and 25 cm. In addition, two voxel sizes were used: 1.9 cm, the finest granularity used for voxel carving in Section 3.1.4, and 2.5 cm, which is a whole-number divider of the supervoxel resolutions and leads to an equal number of voxels assigned to each supervoxel. The compactness parameter of the proposed algorithm was set to $\alpha = 0.9$ to include both spatial distance as well as normal information in the computation time. The number of iterations was set to 10. The experimental system for all evaluations was an Intel Pentium Intel(R) Core(TM) i7-3770 CPU with 3.40 GHz and a NVIDIA GeForce GTX 660 Ti.

Supervoxel Segmentation Table 3.3 shows the runtimes for the presented supervoxel segmentation algorithm for both the CPU and GPU implementation as well as for the PCL [138] implementation of VCCS [121]. The parameters for VCCS were set to match the parameters of the algorithm developed in this work ($\lambda = 0$, $\mu = 1$, $\epsilon = 1$ which is comparable to $\alpha = 0.9$).

First, the improvements of the GPU over the CPU implementation will be discussed. The GPU implementation gives a speedup of up to 15 times (*e.g.*, cube example with supervoxel sizes of 10 cm). In general, the runtime improvements for the GPU implementation relative to the CPU implementation become better for a higher number of supervoxels or, equivalently, smaller supervoxel sizes. The reasons are twofold. First, the GPU implementation requires a certain number of parallel tasks for maximum efficiency. Second, the number of boundary voxels is larger for smaller supervoxels which is also in favor for the GPU implementation that processes these boundary voxels in parallel.

VCCS [121] achieves good runtimes and scales well with the number of supervoxels. Compared to the CPU version of the presented algorithm, it is in particular faster for examples with smaller supervoxel numbers, as is the case for the UMPM example. With an increasing number of supervoxels, however, the CPU version is slightly faster and both are comparable to each other for the worst case scenario. Compared with the GPU version of the presented algorithm, VCCS [121] is generally slower. While the speedup for the UMPM example is at most 4 times, it becomes larger for an increasing number of supervoxels with a speedup between 5 and 20 times for the cube example. This is important for this work because too large supervoxels will lead to inaccurate results for pose estimation, as Section 4 will show. Therefore, smaller supervoxels are preferable. Also, VCCS [121] does not achieve runtimes that are fast enough to be useful as a preprocessing step for real-time pose estimation as Table 3.3 shows.

Supervoxel Graphs Table 3.4 gives the runtime analysis for the computation of the supervoxel graph and Table 3.5 for the computation of the weighted supervoxel graph. Similar to the supervoxel segmentation, the GPU implementation achieves a significant reduction of computation time compared to the CPU implementation.

The computation times for the cube scenario are worse than for the UMPM scenario because the graph is almost fully connected. This corresponds to the worst case scenario. However, the computation times for graph construction of one person in the UMPM scenario are in the lower one-digit millisecond range. This low computation time is required for this work because the supervoxel graphs will be used to extract body part candidates for real-time pose estimation in Section 4.

The computation times for the weighted supervoxel graph are slightly higher than for the normal supervoxel graph for the UMPM scenario. The reason is the increased number of connections for the fully connected weighted supervoxel graph. For the cube scenario, however, both graphs are (almost) fully connected. Here, the implementation for the normal supervoxel graph, which is optimized for not fully connected graphs, is slower than the weighted supervoxel graph implementation, which is optimized for fully connected graphs.

3.3.5 Conclusion

This section presented an algorithm to compute the supervoxel segmentation of a volumetric voxel representation. In addition, it introduced the concept of the supervoxel graph that builds on top of this volumetric representation. Further, a metric to measure the

compactness of supervoxel segmentations was presented and the correlation with other metrics evaluated.

The presented supervoxel algorithm achieves very good results in comparison to the state-of-the-art, both in terms of segmentation quality as well as runtime. It is therefore well suited as a preprocessing step for pose estimation. By providing the supervoxel segmentation and graph computation in real-time, the number of voxels, and therefore the search space, can be effectively reduced for human pose estimation and body tracking that will be presented in the next section.

Table 3.3: Supervoxel segmentation runtimes. The table shows runtimes for the presented algorithm (both CPU and GPU implementations) as well as for VCCS [121]. Two voxel sizes and four different supervoxel sizes were used with four scenarios: a synthetic cube with side lengths 1 m, one example frame of the UMPM dataset [162] showing a single person, and best and worst case scenarios with empty and full voxel grids.

Size	Supervoxel	Scenario	Voxel size: 1.9 cm			Voxel size: 2.5 cm		
$[cm]$	number		CPU $[ms]$	GPU $[ms]$	VCCS $[ms]$	CPU $[ms]$	GPU $[ms]$	VCCS $[ms]$
10	1519	Cube	741.4	43.3	885.9	371.9	23.5	356.4
10	361	UMPM	186.6	16.2	70.9	97.9	9.8	27.1
10	0	Best case	69.5	2.0	0.0	41.5	1.4	0.0
10	56259	Worst case	26419.6	1993.2	24319.8	13375.7	878.4	9682.5
15	506	Cube	646.4	64.3	995.9	309.0	32.0	387.1
15	154	UMPM	159.0	29.5	80.3	77.4	16.9	28.8
15	0	Best case	52.9	1.7	0.0	25.7	1.1	0.0
15	16407	Worst case	21947.4	1989.8	27655.3	10638.9	870.4	10595.0
20	238	Cube	610.5	106.2	1089.2	286.1	45.4	428.0
20	82	UMPM	149.9	64.5	85.8	70.2	29.3	30.9
20	0	Best case	50.9	1.7	0.0	23.7	1.0	0.0
20	7411	Worst case	21230.9	2100.0	29484.1	10233.5	864.9	11549.7
25	148	Cube	650.1	188.3	1202.7	302.9	88.6	470.5
25	48	UMPM	142.9	119.2	94.4	63.8	60.3	33.4
25	0	Best case	50.1	1.6	0.0	22.9	1.0	0.0
25	3543	Worst case	18814.8	1979.0	33067.1	8789.2	851.7	12800.9

Table 3.4: Supervoxel graph runtimes. The table shows runtimes to compute the supervoxel graph for two voxel and four supervoxel sizes, both for the CPU as well as the GPU implementation. Two scenarios were used: a synthetic cube with side lengths 1 m and one example frame of the UMPM dataset [162] showing a single person.

Size [cm]	Supervoxel number	Scenario	Voxel size: 1.9 cm		Voxel size: 2.5 cm	
			CPU [ms]	GPU [ms]	CPU [ms]	GPU [ms]
10	1519	Cube	2157.3	112.7	1650.8	82.2
10	361	UMPM	36.7	2.8	27.1	2.1
15	506	Cube	240.7	15.0	186.1	10.8
15	154	UMPM	7.4	0.6	5.5	0.6
20	238	Cube	55.8	3.4	42.6	2.5
20	82	UMPM	2.6	0.3	1.9	0.3
25	148	Cube	24.4	1.4	17.2	0.9
25	48	UMPM	1.1	0.4	0.7	0.4

Table 3.5: Weighted supervoxel graph runtimes. The table shows runtimes to compute the weighted supervoxel graph for two voxel and four supervoxel sizes, both for the CPU as well as the GPU implementation. Two scenarios were used: a synthetic cube with side lengths 1 m and one example frame of the UMPM dataset [162] showing a single person.

Size [cm]	Supervoxel number	Scenario	Voxel size: 1.9 cm		Voxel size: 2.5 cm	
			CPU [ms]	GPU [ms]	CPU [ms]	GPU [ms]
10	1519	Cube	1675.6	95.1	1199.0	69.1
10	361	UMPM	106.2	5.5	71.5	3.0
15	506	Cube	184.3	12.6	135.8	8.8
15	154	UMPM	21.0	0.9	13.9	0.5
20	238	Cube	43.3	2.8	31.3	2.0
20	82	UMPM	6.4	0.4	4.1	0.3
25	148	Cube	18.5	1.1	12.7	0.8
25	48	UMPM	2.6	0.4	1.6	0.3

4 Pose Estimation and Body Tracking

This section introduces the pose estimation and body tracking algorithm developed in this thesis. The main contribution described here is an investigation of how the search space and computational complexity of pose estimation can be reduced by using segmentation as a preprocessing step. This is demonstrated with the example of estimating the high-dimensional configuration of a human pose in real-time by using supervoxels as building blocks.

The presented system requires only two parameters to be fully specified and is computationally very efficient. Further, it does not rely on large amounts of training data, but extracts all relevant information directly from observed data with little prior knowledge. These characteristics make it very useful for real-world applications.

The pose estimation approach described here combines all elements introduced in the previous sections. It works directly on voxels that were computed with voxel carving as presented in Section 3.1. Therefore, it can be used with different sensor types, like multi-view video cameras and one or more depth sensors, and is thus applicable for a wide range of environments. Motivated by superpixels presented in Section 3.2, the number of voxels and, consequently, the number of input elements is reduced by grouping through supervoxel segmentation, as presented in Section 3.3. Further, body parts are directly sampled from the supervoxel graph introduced in Section 3.3.3.

Figure 4.1 gives an overview of the whole pose estimation process. The volume of the voxel reconstruction is shown in Figure 4.1a and the supervoxel segmentation in Figure 4.1b with the supervoxel graph in Figure 4.1c. The remaining parts will be explained in the following sections: First, the body model and the procedure to sample body parts will be introduced in Section 4.1. Then, the pose estimation system, that is based on the pictorial structures framework, will be explained in Section 4.2. Following the frame-based pose estimation, the extension to the temporal domain for articulated body tracking will be introduced in Section 4.3 including a complexity analysis in Section 4.4. After the system has been introduced, the evaluation on two datasets will be presented in Section 4.5 before concluding in Section 4.6.

4.1 Supervoxel Body Model

This section introduces the body model based on supervoxels that will be used in this work. The body model is represented by a skeleton with twelve rigid limbs that are connected by 15 joints. A visualization of the body model is shown in Figure 4.2.

Estimating the configuration of such a model in 3D is a highly complex task. Each joint location is fully specified by three coordinates leading to a total of 45 degrees of freedom that must be estimated. In principle, each 3D point is a potential candidate for the position of a joint leading to an intractable problem. However, by using supervoxels, this search space can be significantly reduced.

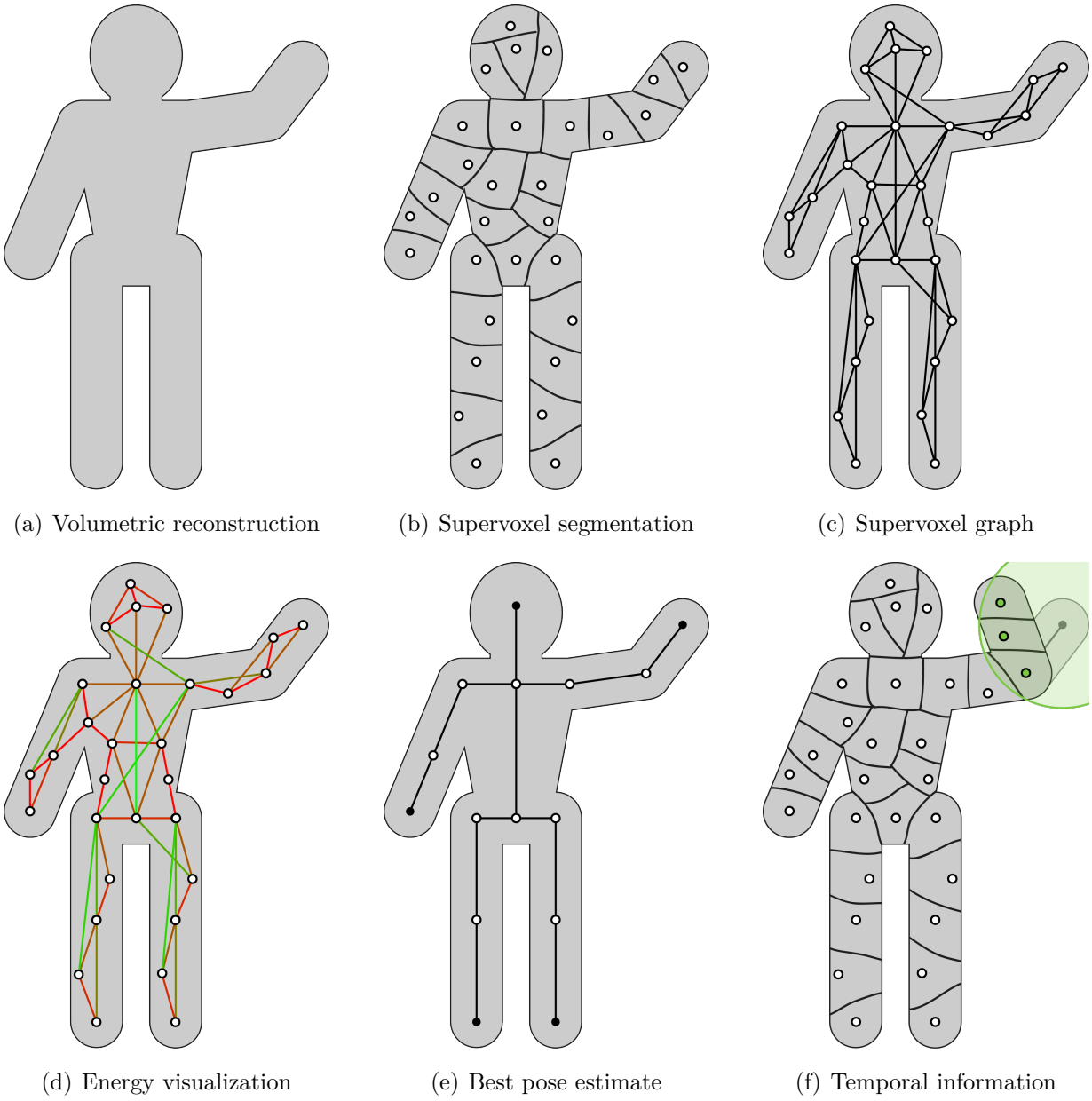
Motivated by Mori's 2D pose estimation with superpixels in [113], the key idea in the presented pose estimation approach is to reduce the search space by limiting joint positions to centers of supervoxels. Thereby, the continuous 3D search space for each joint position is limited to a significantly smaller number of supervoxels.

The search space can be further decreased by reducing the number of potential limbs. With the assumption that joint positions are constrained to supervoxel centers, the rigid limbs are then implicitly restricted to connections between supervoxel centers. However, not any arbitrary connection between two supervoxels is a valid limb candidate. Because limbs are inside the body, the same must also hold for connections between supervoxels. This is in accordance with the definition of the supervoxel graph presented in Section 3.3.3. Therefore, valid body part candidates can directly be sampled from the set of edges of the supervoxel graph.

In summary, by restricting joint positions to centers of supervoxels and by further restricting limbs to connections of the supervoxel graph, the space of valid body configurations is significantly reduced. As the evaluation in Section 4.5 will show, this will later allow for real-time pose estimation. But first, the next section will introduce an efficient estimation algorithm based on pictorial structures.

4.2 3D Pictorial Structures with Supervoxels

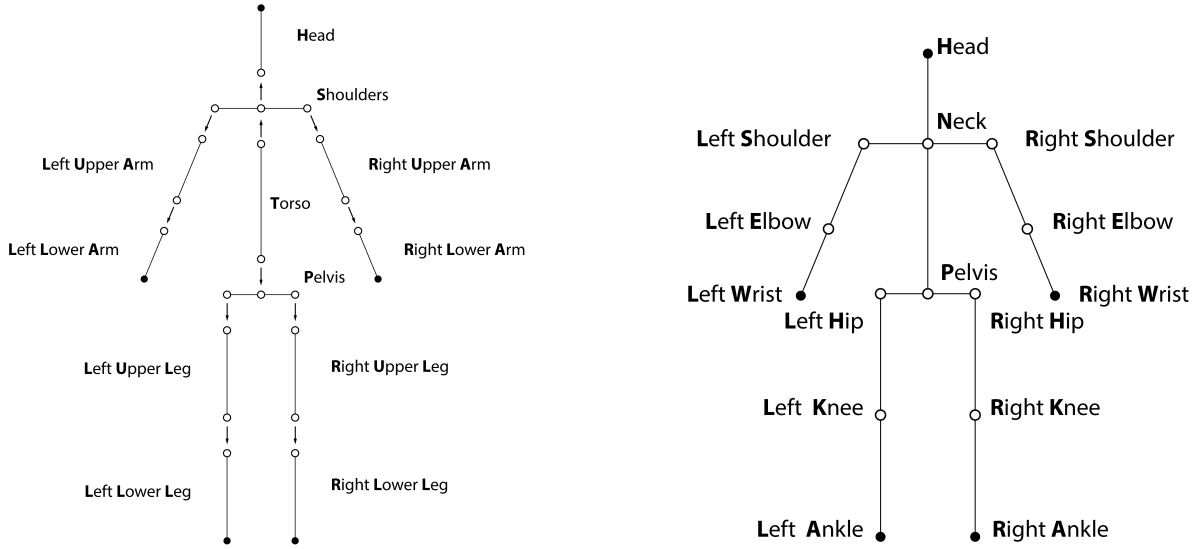
This section introduces the pictorial structures approach for 3D human pose estimation with supervoxels. It is based on the work of Felzenszwalb and Huttenlocher [64] and motivated in part by Burenius *et al.* [45] who presented one of the few efficient approaches for pictorial structures in 3D. Their algorithm uses a discretization of the search space



(a) Volumetric reconstruction (b) Supervoxel segmentation (c) Supervoxel graph

(d) Energy visualization (e) Best pose estimate (f) Temporal information

Figure 4.1: Pose estimation and body tracking process overview. The images show all steps of the pose estimation and body tracking system. (a) The bounding volume is computed with voxel carving and then (b) segmented into supervoxels with the circles representing their centers. (c) The supervoxel graph connects supervoxels if the connection is inside the volume (not all connections are shown). (d) Depending on the lengths of graph edges, their fitness to represent body parts is measured (here the torso fitness is shown with green representing the best fit). (e) The final pose is estimated by minimizing the overall energy of all parts and connections. (f) Previous estimates can be propagated through time and used as additional weights for body tracking.



(a) Kinematic tree

(b) Connecting joints

Figure 4.2: Supervoxel Body Model. The body model consists of twelve rigid body parts, the limbs, that are connected by ten joints. The circles are supervoxel centers. Black circles indicate end joints and white circles indicate connecting joints. Connections between supervoxels are limbs. (a) shows the kinematic tree with limbs as nodes and arrows as directed edges and (b) shows the connecting joint names.

and certain constraints to reduce the overall complexity. In contrast to their work, the approach presented in the following sections shows how pictorial structures can be solved even more efficiently when using supervoxels as building blocks.

Following Felzenszwalb and Huttenlocher [64], as explained in Section 2.3.4, the problem of pose estimation can be formulated as an energy minimization problem. Given a body model θ and a configuration L that consists of N body parts $l_1, l_2, \cdots, l_N$, the task is to find the optimal configuration L^* that minimizes a given energy function. The energy function measures the quality of a configuration based on two parts. First, the unary appearance term m_i evaluates the appearance of body parts i represented by l_i. Second, the binary term d_{ij} evaluates the connection between two parts l_i and l_j, while connections are defined by the body model and represented by set E. The optimal configuration minimizes the overall energy:

$$L^* = \underset{L}{\operatorname{argmin}}\Big(\sum_{i}^{N} m_i(l_i) + \sum_{(i,j)\in E} d_{ij}(l_i, l_j)\Big). \tag{4.1}$$

As explained in Section 2.3.4, the energy minimization originates from a statistical formulation. The energy terms are the result of taking the negative logarithm over the respective probability distributions. Given an expected mean value $\bar{x}$ and standard deviation σ, they follow the general form

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-0.5\frac{(x-\bar{x})^2}{\sigma^2}}. \tag{4.2}$$

Therefore, by taking the negative logarithm over the square root of this function and further simplifying by removing constant parts, the energy terms are of the form

$$m(x) = \frac{|x - \bar{x}|}{\sigma} \tag{4.3}$$

While the distributions are chained with multiplication, the energy terms are chained by summation. The same holds for distributions that consist of more than one term. Now, the specific energy terms will be explained in more detail, starting with the unary term.

4.2.1 Evaluating Parts

This section introduces the evaluation of the unary energy term m_i of Equation 4.1. This term is also called the data term because it measures how well a specific part l_i fits to the

data, *i.e.*, the observation. In other approaches, the appearance term is usually trained with annotated ground truth data to model the appearances of single parts. However, this implicitly introduces a bias because new observations must fit to the training data. For example, the part detector in [146] uses relative offsets in specific directions. Therefore, they do not work anymore if the camera is rotated. In this work, the appearance term will be evaluated with only little prior knowledge about part appearances and no explicit training, thus being more generally applicable.

Here, body parts l_i are directly sampled from the 3D supervoxel graph. As its connections are inside the segmented voxel volume, it is already guaranteed that they are valid candidates. Therefore, only the lengths of these connections have an influence on the appearance term. The more similar the connection lengths are to the expected body part lengths, the better is their match and, consequently, the lower is their energy. The only required information are the expected lengths of the respective body parts. These can easily be computed by using anthropometric ratios.

In anthropometrics, one task is to measure the limb lengths of people and to provide general ratios for body part sizes compared to the overall posture height. Figure 4.3 shows an example of a body model with ratios from Drillis and Contini [60] cited by Fromuth and Parkinson [68]. These ratios allow to compute the expected sizes for all limbs with just an estimate of the overall posture height. For example, if a person's height is 1.8 m, their expected shoulder width is $1.8\ \text{m} \cdot 0.259 \approx 0.47\ \text{m}$. The overall posture height can either be assumed to be given or directly estimated by taking the largest expansion of the observed data. In case of calibrated cameras and a fixed ground plane, this can even further be simplified by just taking the maximum height from the ground plane.

It will now be explained how body parts are sampled from both the strict as well as the weighted supervoxel graph and how the corresponding graph energies are computed.

Sampling from the Supervoxel Graph

The body parts l_i are directly sampled from the supervoxel graph, as was explained above. This means that every connection is a candidate for each body part. To measure how well a connection represents a part, its length is compared to the expected part lengths that are given by applying the anthropometric ratios. The unary energy term can then directly

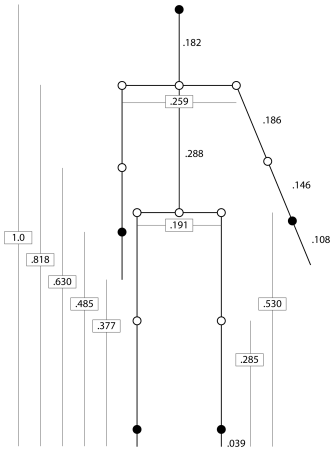


Figure 4.3: Anthropometric ratios. Assuming a total posture height of 1, the lengths of the body parts are given as fractions. The ratios are from [60] and the drawing is based on [68].

be computed by comparing the length of a body part, given by $||l_i||$, with the expected length $||\hat{l}_i||$:

$$m_i^a(l_i) = \frac{\left| ||l_i|| - ||\hat{l}_i|| \right|}{||\hat{l}_i||}. \tag{4.4}$$

Equation 4.4 is normalized by the expected length. This will also be done for all following energy terms to allow for a comparison between them without the need of additional weighting.

Equation 4.4 is evaluated for all connections of the supervoxel graph and for all N body parts. This can be done very efficiently because only lengths must be compared. It can also be parallelized because all limbs can be evaluated independently. As visual reference, Figure 4.1d shows color-coded energies of connections, in this case for the torso.

Sampling from the Weighted Supervoxel Graph

The previous section described sampling from the strict supervoxel graph where all edges reside fully within the segmented volume. As explained in Section 3.3.3, this strict formulation can lead to the removal of correct body part candidates due to errors in the segmentation. If only one voxel on a connection was erroneously carved, the whole

connection is discarded. Therefore, the weighted supervoxel graph provides additional robustness through weighted edges that indicate what fraction resides inside the volume.

The weights provided by the weighted supervoxel graph can either be used as an unary energy term similar to Equation 4.4 with an expected value of 1 or by filtering edges that fall below a certain threshold. When using the weight directly as energy, the problem is that it is measured with a different unit than body lengths and, thus, normalization is more difficult. Therefore, this work follows the second approach and filters edges that fall below a certain threshold γ. Let $w(l_i)$ give the weight of body part l_i. Then, the modified unary term is given by

$$m_i^\tau(l_i) = \begin{cases} \frac{\left| ||l_i|| - ||\hat{l}_i|| \right|}{||\hat{l}_i||} & \text{if } w(l_i) \geq \tau \\ \infty & \text{otherwise} \end{cases} \tag{4.5}$$

In the remainder of this work, Equation 4.5 will be used to measure part appearances, *i.e.*, $m_i = m_i^\tau$, if not specified otherwise. In case the strict supervoxel graph is used, Equation 4.5 is equivalent to Equation 4.4 by setting the threshold to $\tau = 1$.

The unary term is also well suited to model additional knowledge about part positions, *e.g.*, given by part detectors, and also to include temporal information for body tracking. Both extensions will be addressed in Section 4.3.

This concludes the section about the unary energy term that measures part appearances. It can directly be computed given the data with very little prior knowledge and it works with both strict and weighted supervoxel graphs. The next section will explain how the connections between parts are evaluated.

4.2.2 Evaluating Connections

This section explains how connections between parts are evaluated given the body model. Evaluating all possible connections is one of the computationally more expensive parts when using pictorial structures. By using supervoxels, however, the number of possible connections can be significantly reduced. Also, evaluating connections requires modeling the pose prior which is usually done through training with labeled ground truth data. In this work, the prior is modeled directly and merely expresses that parts should not overlap as will now be explained.

Restricting Possible Connections

In principle, each pair of body parts can be connected. This is in particular the case if connecting joints are modeled with spring-like connections that allow a certain variability in the relative positions of joints. This leads to a large number of possible connections that must be evaluated when searching for the best configuration. By using supervoxels, this search space can be drastically reduced.

At the beginning of this section, one key assumption of this approach was explained: joint positions are restricted to supervoxel centers. Therefore, two body parts that are defined by their starting and end joints can only be connected if they share the same supervoxel as their connecting joint. Let $c_{ij}(l)$ give the supervoxel at the connecting joint of parts i and j. Then, the binary term d^s, that restricts possible connections based on supervoxels, is given by

$$d^s_{ij}(l_i, l_j) = \begin{cases} 0 & \text{if } c_{ij}(l_j) = c_{ij}(l_i) \\ \infty & \text{otherwise} \end{cases} \tag{4.6}$$

With Equation 4.6, it is possible to remove a large number of connections that do not share the same connecting supervoxel, thereby reducing the overall complexity. However, as long as a connection exists between two parts, its energy is uniform. The next section shows how this can be modified in order to consider overlap between parts as well.

Evaluating Part Overlap

When using a skeleton model for pose estimation with volumetric data, it is particularly important to ensure that parts do not overlap. Otherwise, the skeleton model could theoretically be folded very tightly to fit into almost arbitrarily small volumes because the skeleton itself has no volumetric expansion. Other approaches address this problem, for example, by using learned joint angle distributions or volumetric primitives. Then, however, it must still be ensured that body parts cannot overlap. Depending on the volumetric primitives used, this can be nontrivial and computationally expensive. Further, such primitives additionally introduce the problem of how to initialize their parameters.

This work follows a rather direct approach to ensure that parts do not overlap with each other. As the evaluation in Section 4.5 will show, this is sufficient to achieve state-of-the-art results on challenging datasets. The basic assumption is that connected body parts overlap the least if they are fully extended.

Let $d^e_{ij}(l_i, l_j)$ be the energy that measures the overlap between body parts l_i and l_j. Let $g_{ij}(l_i, l_j)$ give the distance between the end joint of l_j to the end joint of l_i. Further, let $\hat{g}_{ij}(l_i, l_j)$ be the expected distance between their end joints if both limbs would be perfectly extended. Then, the energy term is given by

$$d^e_{ij}(l_i, l_j) = \frac{1}{\frac{g_{ij}(l_i,l_j)}{\hat{g}_{ij}(l_i,l_j)}} = \frac{\hat{g}_{ij}(l_i, l_j)}{g_{ij}(l_i, l_j)}. \tag{4.7}$$

It is also possible to define the part overlap score to simultaneously consider multiple parts, if they are already known. This ensures that all parts of the skeleton have a minimal overlap and not only connected parts. Let L be the configuration with parts l_i and let all parts $j < i$ be known. Then,

$$d^e_i(l_i, L) = \sum_{k=1}^{j} d^e_{ik}(l_i, l_k). \tag{4.8}$$

The binary term models the pose prior. By preferring non-overlapping parts as described above, the prior favors straight poses. However, this bias is relaxed by enforcing that body parts must be inside the volumetric reconstruction through the supervoxel graph. If the volume is too small, *e.g.*, because the legs are bent, straight poses do not fit into it and, therefore, the next best pose, *e.g.*, with bent legs, is preferred.

This concludes the sections about the unary and binary energy terms. The next section shows how these terms are used in an algorithm to compute the optimal pose configuration.

4.2.3 Pose Estimation Algorithm

Given a set of parts that were sampled from the supervoxel graph and both unary and binary energy terms to evaluate these parts and their connections, all required information is available to compute the optimal pose. Because the body model follows a tree structure with no cycles and connections between parts can easily be computed, the overall estimation can be solved with dynamic programming [64]. Here, the min-sum algorithm is used with an algorithm layout similar to [45].

The min-sum algorithm follows the dynamic programming strategy by breaking down the complex problem of estimating a human pose into less complex subproblems. Here, the subproblems are finding pairs of connected limbs. The estimation is separated into two phases that follow a message passing pattern. During the first phase, messages, *i.e.*, accumulated energies, are passed up starting from the leaf nodes up to the root. Then, the

best root, *i.e.*, the one with the lowest accumulated energy, is selected. In the following second phase, messages are passed down starting from the selected root node and the optimal child nodes are successively selected. Both parts will now be explained in more detail. Algorithm 4 shows pseudo code for this procedure.

Algorithm 4 Min-sum pose estimation with supervoxels

Input: supervoxel graph $G = (\mathcal{V}, E)$
Output: pose configuration L^*
Define $p(i)$: return parent node of body part i

// Initialization
$\forall l \in E \; \forall n \in \{1, \cdots, N\}$: initialize part scores $m_n(l_n)$
// Passing messages up
for $n := N$ *to* 2 **do**
  for $l_{p(n)} \in E$ **do**
    $\overline{m} := \min_{l_n}(d^s_{n,p(a)}(l_n, l_{p(n)}) + m_n(l_n))$
    $m_{p(n)} := m_{p(n)} + \overline{m}$
  end for
end for
// Passing messages down
$l_1^* := \operatorname{argmin}_{l_1}(m_1(l_1))$
$L^*(1) := l_1^*$
for $n := 2$ *to* N **do**
  $l_n^* := \operatorname{argmin}_{l_n}(d^e_n(l_n, L^*) + m_n(l_n))$
  $L^*(n) := l_n^*$
end for

Up Path

During the first part, messages are passed up from child nodes to their respective parents. The paths are specified by connections in the body model. For example, at the beginning, messages are passed from all lower arm parts to all upper arm parts. The messages represent energies and combine both part appearances of child nodes as well as connections between child and parent nodes.

For each parent node, only one message is accepted from its child nodes, namely the one with the lowest energy, and added to its own energy. The reason is as follows. If this parent

node would be selected as body part for the final pose estimate, it would select the child with minimum energy. By adding the child's energy to its own, this information is thereby propagated to the parent's parent and available when it selects its optimal child node.

At the end, the accumulated energies of the root node represent the overall energy of a pose if only the best children are selected starting from this root node. Therefore, the optimal root node is the one with the lowest accumulated energy. Here, the root node of the body model is the torso. This is the starting point for the down path.

Down Path

Starting from the selected torso, *i.e.*, the root node of a pose hypothesis, messages are passed down. Similar to the up path, messages represent energies. They are given by the (accumulated) energy of the child node and the energy of the connection between child and parent. On the down path, the children are selected that have a minimal overall energy. This is significantly faster than the up path because now, one part is already fixed for each connection.

In the original pictorial structures approach, the binary term evaluating connections between parts is the same for both parts. Here, different terms are used. During the up path, the binary term in Equation 4.7 is used that gives a uniform energy as long as parts are connected. This ensures that the optimal parts are selected based on their appearance terms. During the down path, the binary term of Equation 4.8 is used that also models part overlap. The reason is that only during the down path the overall overlap between all parts can be computed. During the up path, this would only be possible for connected parts. The result of this procedure is the final pose estimate for the current time step. Figure 4.1e shows an example of a selected pose.

In general, the min-sum algorithm computes the optimal pose. Due to the two different binary energy terms used in this work, the pose estimation becomes a greedy algorithm and there is the potential for suboptimal solutions. However, sampling only a small number of poses (Section 4.3.4) is sufficient for stable results, as the evaluation will show.

The algorithm presented here estimates the pose solely based on information available in the current time step. Besides the anthropometric ratios and an estimate of the posture height, it requires no additional prior information. Nevertheless, additional information, *e.g.*, through tracking or detectors, can be seamlessly integrated as the next sections will show.

4.3 Articulated Body Tracking

This section introduces the articulated body tracking algorithm. In particular, it introduces a mechanism for integration of arbitrary information into the pose estimation process by using *supervoxel energies*. This mechanism will be demonstrated with two examples: first, including temporal information to track body parts over time and second, integrating knowledge about part positions available through part detectors.

4.3.1 Supervoxel Energies

This section introduces the concept of supervoxel energies. It can be used to integrate available information about the pose directly on the level of supervoxels. These supervoxel energies are then used as additional unary energy terms in the pose estimation algorithm introduced in Section 4.2.3 without any modifications. This demonstrates that supervoxels not only reduce the search space for pose estimation, but also allow to incorporate pose knowledge.

The *supervoxel energy* expresses how well a supervoxel is suited to represent a certain joint. It is a general concept and there are various ways to compute specific supervoxel energies, as the next two sections will show. The purpose of supervoxel energies is that they directly affect the unary energy terms for body parts. This means that body parts with two supervoxels as joints that have a low energy will consequently also have a lower overall energy and, thus, be preferred during pose estimation.

The supervoxel energies are propagated to limbs by averaging them. Let k be a joint in the body model ϕ and let V be a supervoxel. The energy of supervoxel V for joint k is then given by $\varphi_k(V)$. Further, let body part l_i have start joint $s(i)$ and end joint $e(i)$ and let $s(l_i)$ and $e(l_i)$ give the supervoxels at the start and end joints, respectively. Then, the unary supervoxel energy $m_i^s(l_i)$ combines both supervoxel energies and is given by

$$m_i^s(l_i) = \frac{\varphi_{s(i)}(s(l_i)) + \varphi_{e(i)}(e(l_i))}{2}. \tag{4.9}$$

In the case that supervoxel energies are used, the unary term is given as a combination of Equations 4.5 and 4.9:

$$m_i(l_i) = m_i^\tau(l_i) + m_i^s(l_i). \tag{4.10}$$

The advantage of supervoxel energies is that additional information can be modeled on supervoxel level without modifications of the overall algorithm. The next two sections will show how the concept of supervoxel energies can be used to track body parts with temporal pictorial structures and how to included part detectors.

4.3.2 3D Temporal Pictorial Structures

Frame-based pose estimation, as described above, uses only information available in the current time step. In particular, estimation starts anew without knowledge about previous poses. In contrast, body tracking relies on past information, in particular about the previous poses. Typically, tracking approaches, like the Kalman filter or particle filters, consist of two steps: first, based on previously observed poses, the new pose for the current frame is predicted. Then, this prediction is adjusted based on currently observed features. One drawback of such approaches is that they fail if their tracks get stuck in a local minimum or are lost. For these cases, additional mechanisms for detection of tracking failures and recovery are required.

The tracking algorithm described in this section follows a different approach. Instead of utilizing a separate mechanism for tracking, it integrates temporal information directly into pose estimation using the concept of supervoxel energies. Previously estimated poses propagate their energies to supervoxels in the current time step if they are close enough. The better the pose, the lower is its energy and, consequently, the supervoxel energy. Therefore, these supervoxels will be preferred when computing the current pose estimate.

Let $\mathcal{L}_{s:t} = \{L_s, \cdots, L_t\}$ be optimal configurations, *i.e.*, pose estimates, from previous time steps s to t. Let k be a joint and let V be a supervoxel of the current supervoxel segmentation $\mathcal{V}$. Further, let $\varphi_k(L)$ give the minimum energy of body parts $l \in L$ that share joint k and let $j_k(L)$ give the 3D joint coordinates, *i.e.*, the supervoxel center, of joint k. Joints of past poses propagate their energies to supervoxels if their distance $d(j_k(L), V)$ is below threshold Δ. Then, the *supervoxel tracking energy* is

$$\varphi_k^t(V) = \begin{cases} \min\limits_{L \in \mathcal{L}_{s:t}} \varphi_k(L), & \text{if } \exists L \in \mathcal{L}_{s:t} : d(j_k(L), V) < \Delta \\ \max\limits_{L \in \mathcal{L}_{s:t}} \varphi_k(L), & \text{if } \nexists L \in \mathcal{L}_{s:t} : d(j_k(L), V) < \Delta \end{cases}. \tag{4.11}$$

Equation 4.11 basically states that, for each joint, each supervoxel is assigned the minimum energy of limbs from previous poses if their corresponding joints are sufficiently close. If no

such pose exists, the maximum limb energy with respect to this joint is assigned. Figure 4.1f visualizes this assignment for an arm joint.

Limbs are connections between two supervoxels. Therefore, the energies of single supervoxels have to be combined for body parts. The unary tracking energy $m_i^t(l_i)$ for part l_i is then given analog to Equation 4.9 by

$$m_i^t(l_i) = \frac{\varphi_{s(i)}^t(s(l_i)) + \varphi_{e(i)}^t(e(l_i))}{2} \tag{4.12}$$

and the final unary energy term by

$$m_i(l_i) = m_i^\tau(l_i) + m_i^t(l_i). \tag{4.13}$$

Due to the modified unary energy term in Equation 4.13, limbs that are close to their corresponding positions in past time steps will be preferred in the next time step. This has the advantage that pose estimation and tracking are not two separate steps but are combined in the same approach. Further, getting stuck in local minima is avoided because then all supervoxels would get an equally low energy resulting in no preference at all.

4.3.3 Integrating Part Detectors

Part detectors or appearance-based descriptors are popular for pose estimation because they reduce the number of positions of potential body parts and, therefore, the space of possible poses, as was also discussed in Section 2.3. The approach described here does not require such detectors because it achieves a search space reduction through supervoxel segmentation. However, this section shows how such information can be directly integrated, if it is available.

The mechanism follows the concept of supervoxel energies presented in Section 4.3.1. It can either use an energy based on the appearance similarity or a binary score to activate or deactivate supervoxels as joint candidates. These are just two examples and there are no limitations of how such an energy can be computed. Assuming detector energy $\varphi_k^d(V)$ is given for supervoxel V and joint k, the unary limb energy term $m_i^d(l_i)$ is computed as in Equation 4.9. An evaluation with a simulated part detector will be shown in Section 4.5.

This concludes this section that has shown how both temporal as well as further available information about part positions can be directly integrated into the supervoxel-based pose estimation framework.

4.3.4 Sampling Poses

The min-sum algorithm computes a pose estimation for the given observation. However, it is sometimes preferable to sample more than one pose hypothesis for a given frame. In this work, multiple hypotheses are sampled by taking multiple paths for the down phase of Algorithm 4.

As was explained in Section 4.2.3, the down path and, consequently, the pose is determined by the selected torso. Therefore, by selecting different torsos, different paths are taken which results in various pose estimates. To ensure good starting hypotheses for the down path, torsos are sampled by selecting the ones with the lowest accumulated overall energies.

Due to the modified binary energy term d_i^e that is used during the down path, the torso with the lowest energy does not necessarily result in the optimal pose. Therefore, given all pose hypotheses after completing all down paths, the one with the lowest energy is chosen as the final result of the pose estimation process.

4.4 Complexity Analysis

This section analyzes the complexity of the presented pose estimation algorithm, both for time as well as space complexity.

Time Complexity Regarding time complexity, both message passing phases of Algorithm 4 as well as initialization must be considered.

Let N be the number of body parts and $|\mathcal{V}|$ the number of supervoxels. Then, the number of potential body parts, *i.e.*, connections of the supervoxel graph, is in the worst case $|\mathcal{V}|^2$. It then follows that initialization is in $\mathcal{O}(|\mathcal{V}|^2)$.

The up phase is generally the computationally most expensive one. Other approaches typically require that the connections between all pairs of body parts are evaluated. For the presented algorithm, this would result in a time complexity of $\mathcal{O}(|\mathcal{V}|^4)$. However, by enforcing that connected body parts must share the same supervoxel as connecting joint, this complexity is reduced to $\mathcal{O}(|\mathcal{V}|^3)$. This is one key characteristic that leads to the efficiency of the presented algorithm.

The down phase is computationally significantly less expensive than the up phase. The reason is that for each connection, one body part including the connecting joint has already been fixed resulting in a complexity in $\mathcal{O}(|\mathcal{V}|)$.

The additional use of supervoxel energies, *e.g.*, for tracking, does not change the overall complexity class. The information must only be propagated to supervoxels. Therefore, the complexity is in $\mathcal{O}(|\mathcal{V}|)$.

The overall time complexity of the pose estimation algorithm is determined by the up phase and therefore in $\mathcal{O}(|\mathcal{V}|^3)$.

Most similar to the presented algorithm are the two approaches proposed by Burenius *et al.* [45]. They reduced the complexity by a discretization of both body part translations and rotations. Let $|T|$ be the size of the translation space and $|R|$ the number of possible rotations. Then, their two proposed algorithm variations have a complexity of $\mathcal{O}(|T| \cdot |R|)$ and $\mathcal{O}(|T| \cdot |R|^2)$, respectively. For good results, a search space discretization of 32^3 for translations T and between 8^3 and 16^3 for rotations R was required. Assuming that approximately 100 supervoxels are used for pose estimation, the achieved complexity reduction in the presented system compared to [45] is up to several orders of magnitude.

Space Complexity Regarding space complexity, the presented algorithm requires to store the energies for all connections of the supervoxel graph and for all body parts. This leads to a storage complexity in $\mathcal{O}(|\mathcal{V}|^2)$.

As a comparison, the algorithm proposed by Burenius *et al.* [45] has a storage complexity of $\mathcal{O}(|T| \cdot |R|)$. Similar to time complexity, the proposed algorithm based on supervoxels achieves a complexity that is several orders of magnitude lower.

This concludes the introduction of the pose estimation algorithm developed in this thesis. The next section will show an evaluation of this approach.

4.5 Evaluation

This section presents the evaluation of the pose estimation and body tracking algorithm described in the previous sections. It is organized as follows. First, the two datasets used for evaluation and the experimental setup are explained. In order to acquire a better understanding of the impact of the various parameters, they are systematically evaluated for pose estimation. It follows an evaluation of supervoxel energies for body tracking as well as part detectors. Then, an evaluation with synthetic voxels follows before showing qualitative example results of estimated poses. The section concludes with an analysis of runtimes and a comparison to state-of-the-art approaches.

4.5.1 Datasets

This section introduces the two datasets that were used in the evaluation. Both datasets are suitable for pose estimation with multi-view video.

HumanEva-I Dataset

The HumanEva-I dataset was introduced by Sigal and Black in 2006 [148] with an extended discussion by Sigal *et al.* in 2010 [147]. It is one of the commonly used datasets for multi-view human pose estimation. The dataset shows sequences of three actors (S1 - S3) performing various actions. There are three trials for each action: the first contains video data and ground truth for training and validation, the second only contains videos for evaluation, and the third only ground truth (*e.g.*, to train pose priors). Each sequence shows one actor at a time and consists of several hundreds of frames. The action names are *Box* (B), *Gestures* (G), *Jog* (J), *ThrowCatch* (T), and *Walking* (W). Each sequence was recorded with seven calibrated cameras: three color cameras with a resolution of 640×480 pixels and four grayscale cameras with a resolution of 644×484 pixels. Ground truth was measured with a marker-based Vicon system and 3D joint positions are available. Example images are shown in Figure 4.4.

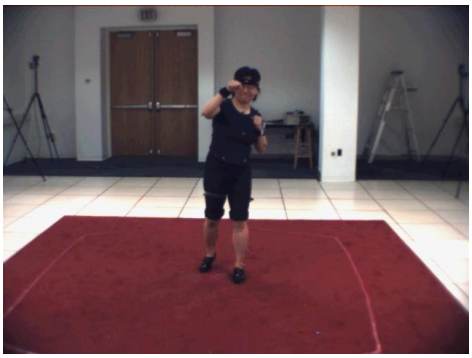
The dataset also contains images to train models for foreground segmentation, but their quality is rather poor. Therefore, foreground images were computed with GrabCut [136] segmentation using a bounding box of the person in combination with ground truth body part locations. Figure 4.5 shows example segmentations.

In this evaluation, all five sequences for actors S1 and S2 were used. As the proposed algorithm requires no training data, all video frames were used for evaluation.

In addition to the HumanEva-I dataset, the HumanEva-II dataset [147, 148] contains an additional *Combo* sequence for two actors (S2 and a new actor, S4), but it was not used in this evaluation as the HumanEva-I dataset provided sufficient data.

Utrecht Multi-Person Motion (UMPM) Benchmark

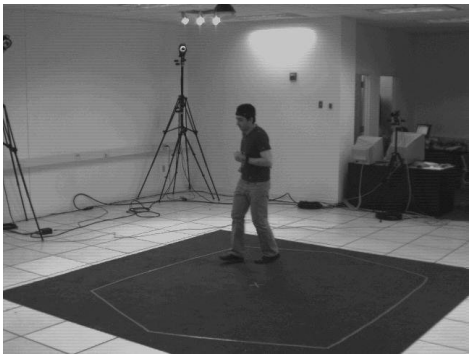
The Utrecht Multi-Person Motion (UMPM) benchmark was presented by Van der Aa *et al.* [162] in 2011. It shows between one to four persons performing various actions. In contrast to the HumanEva dataset, the sequences also contain large objects like chairs and tables that pose an additional challenge. It is a very large dataset with several thousands of frames that were recorded with four calibrated color cameras with a resolution



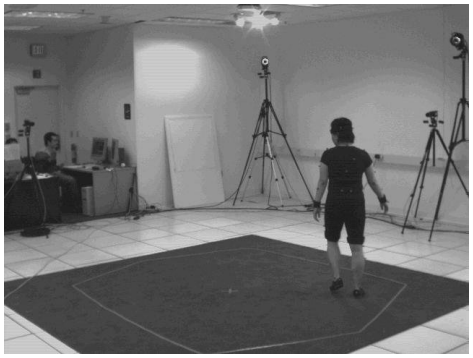
(a) Camera 1, S1 Box

(b) Camera 2, S2 Gestures

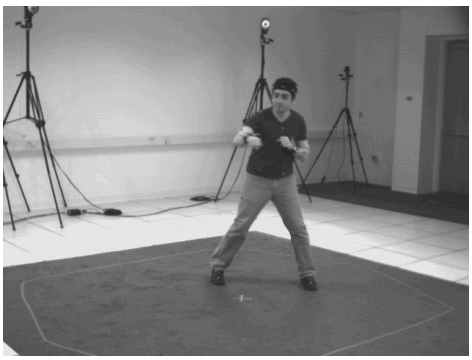
(c) Camera 3, S2 ThrowCatch



(d) Camera 4, S2 Jog



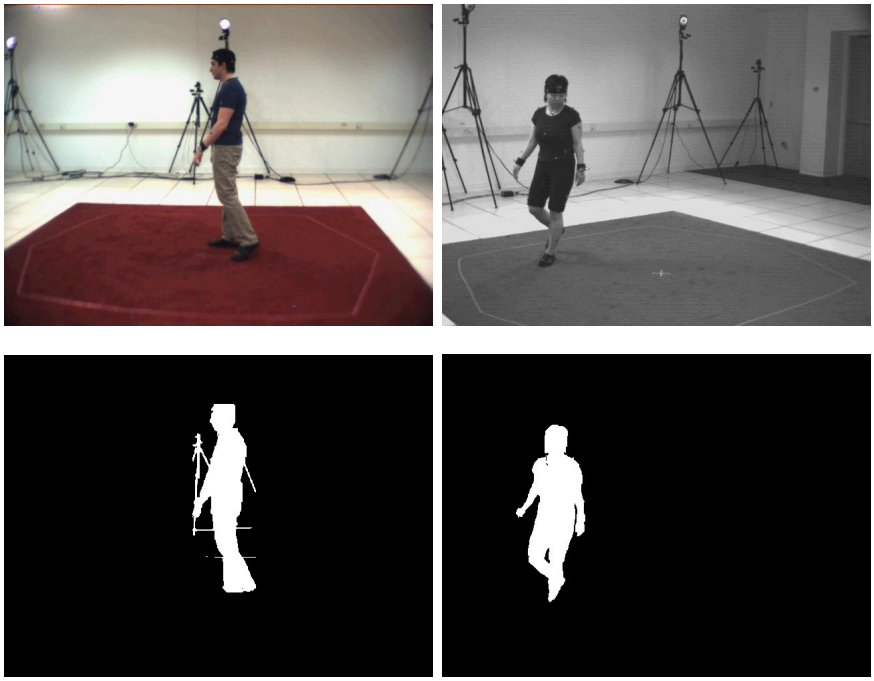
(e) Camera 5, S1 Walking



(f) Camera 6, S2 Box

(g) Camera 7, S1 Gestures

Figure 4.4: HumanEva-I dataset examples. The images show the three color and four grayscale camera views with examples of all five actions performed by actors S1 and S2.



(a) Camera 3, S2 Gestures (b) Camera 6, S1 Walking

Figure 4.5: HumanEva-I foreground segmentations. The images show foreground segmentation results computed with the GrabCut [136] algorithm.

of 644×486 pixels. Ground truth for 3D joint positions was measured with a Vicon system and is available for up to two persons. Figure 4.6 shows example images of the dataset.

The UMPM dataset contains high-quality background images that can be used for foreground segmentation. In this evaluation, a simple background subtraction algorithm trained on the first ten images with a threshold of 25 was used in combination with morphological open and close operators. This leads to errors in the foreground segmentation, especially due to shadows around the person, but resembles quite realistic images that are suitable for evaluation. Figure 4.7 shows example foreground segmentations including errors due to static objects.

In this evaluation, all five sequences with the first person (p1) were used. Each sequence contains approximately $2,500$ frames resulting in a total of more than $12,500$ frames. The sequence names are *Chair* (C), *Grab* (G), *Orthosyn* (O), *Table* (Ta), and *Triangle* (Tr). While the *Orthosyn* and *Triangle* sequences do not contain static objects that occlude the person, the other three sequences do. In addition, the person interacts with these objects, *e.g.*, by sitting on a chair in various poses, grabbing objects, or even lying on a table. The *Orthosyn* sequence mainly shows various arm gestures and the *Triangle* sequence shows walking motions.

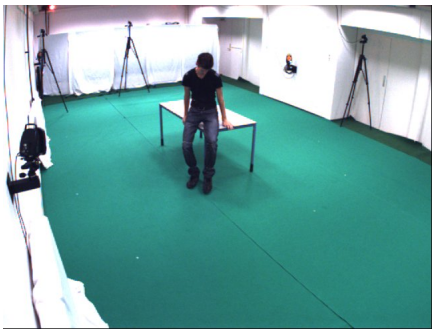
Evaluation Metrics

For evaluation, two metrics were used: the percentage of correct parts (PCP) that was introduced by Ferrari *et al.* [65] and the 3D joint localization error that was introduced Sigal and Black [148] and Sigal *et al.* [147].

Percentage of Correct Parts This metric measures the fraction of correctly estimated parts. A part is counted as correctly classified if the estimated joints at both ends have a distance of at most half the limb length from the corresponding ground truth joints. For example, if a lower arm has a length of 20 cm, both wrist and elbow must be closer than 10 cm to the ground truth joints. This metric has been applied to many pictorial structures approaches as discussed in Section 2.3.4. It is particularly well suited for approaches with supervoxels because it does not penalize inaccuracies that are introduced due to the supervoxel granularity. For comparison to other work, the evaluated body parts include torso, head, upper and lower arms and legs.

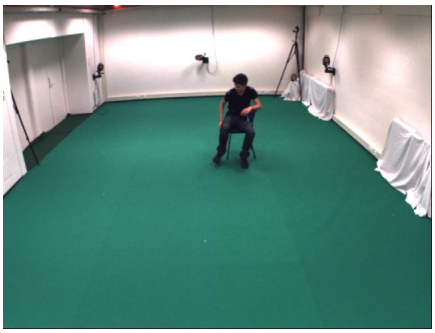
In particular for the UMPM dataset, there is an offset in the ground truth and the skeleton is not always inside the body. As this does directly affect PCP measures, the relative

(a) Camera 1, Orthosyn

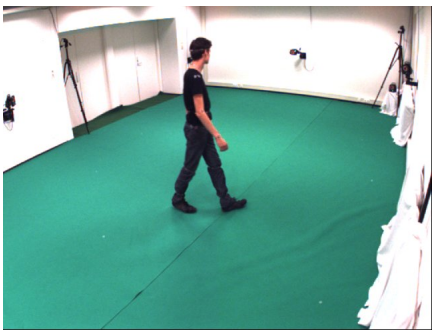


(b) Camera 2, Table

(c) Camera 3, Grab

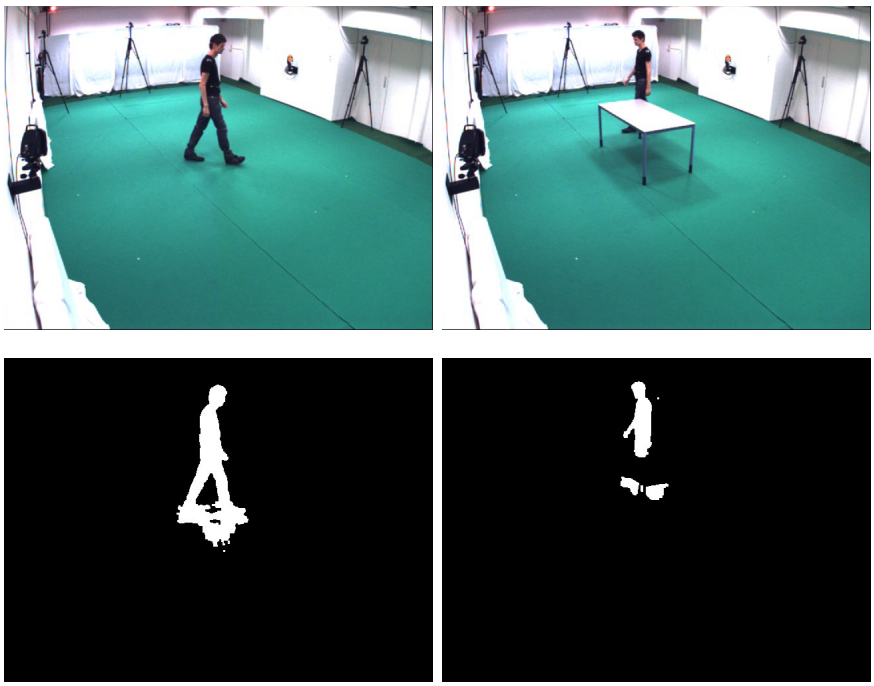


(d) Camera 4, Chair



(e) Camera 3, Triangle

Figure 4.6: Example images of the UMPM dataset. The images show the five action sequences used for evaluation from four camera views.



(a) Camera 2, Triangle (b) Camera 2, Table

Figure 4.7: Foreground segmentation for the UMPM dataset. The foreground segmentations were computed with a basic background subtraction algorithm and post-processed with morphological open and close operators. Due to the simple algorithm, shadows are included in the silhouettes. Further, silhouette defects occur because of static occluders.

comparison of joint positions is used as introduced for the HumanEva dataset in [147] in Equation 6. Here, the root of the body model is the torso center and all differences are computed relative to it.

Evaluation on the UMPM dataset does not always include all sequences. If this is the case, it will be explicitly mentioned. Further, for the three scenes with static occluders, occlusions maps will be used for voxel carving that have been introduced in Section 3.1.2.

3D Joint Localization Error This metric measures the average difference between estimated and ground truth joint positions in millimeters. It was proposed by Sigal and Black [148] and Sigal *et al.* [147] for evaluation with the HumanEva datasets as it allows for a direct comparison between different approaches. Following [147, 148], let L be the estimated pose and $\hat{L}$ the ground truth pose with $j_i(L)$ giving the 3D position of joint i. Let J be the number of joints and let $||\cdot||$ give the Euclidean distance. The 3D error for one frame is then given by

$$D(L, \hat{L}) = \frac{1}{J} \sum_{i=1}^{J} ||j_i(L) - j_i(\hat{L})||. \tag{4.14}$$

The 3D error of a sequence with T frames with poses $L_t \in \mathcal{L}$ and ground truth poses $\hat{L}_t \in \hat{\mathcal{L}}$ then is

$$D(\mathcal{L}, \hat{\mathcal{L}}) = \frac{1}{T} \sum_{t=1}^{T} D(L_t, \hat{L}_t). \tag{4.15}$$

Similar to the PCP metric discussed above, the relative joint localization error is used (Sigal *et al.* [147], Equation 6) with the torso center being the origin of the pose to account for offsets of the marker positions.

Experimental Setup

The same system was used for all experiments, in particular for runtime evaluation. The system was an Intel Pentium Intel(R) Core(TM) i7-3770 CPU with 3.40 GHz with a NVIDIA GeForce GTX 660 Ti for GPU computations. The pose estimation algorithm is available both as CPU as well as GPU implementation.

4.5.2 Pose Estimation Parameter Evaluation

This section presents the evaluation of various parameter settings. One advantage of the presented system is the low number of parameters that are required. The pose estimation algorithm is determined by only two parameters: the threshold τ to filter connections of the supervoxel graph and the number of pose hypotheses K. The preprocessing segmentation algorithms require only four parameters: voxel size c_v, supervoxel size c_V, supervoxel segmentation iterations i, and compactness α. Following the results of supervoxel evaluation in Section 3.3.4, the number of segmentation iterations is fixed to $i = 10$ and the compactness parameter to $\alpha = 1.0$. This leaves a total of only four free parameters for the whole pose estimation algorithm.

The limb sizes in the provided ground truth of the two datasets is different from the anthropometric ratios. Because the PCP measure is based on these part sizes, even correctly estimated poses would then not always be counted as correctly classified. Therefore, the ground truth limb sizes were used for initialization instead of anthropometric ratios.

The evaluation includes parameter values in the following ranges. The voxel sizes c_v are between 2 cm and 4 cm and the supervoxel sizes between 10 cm and 20 cm. The weight threshold for supervoxel graph connections varies between 0.1 and 1 and the number of pose hypotheses is between 1 and 100.

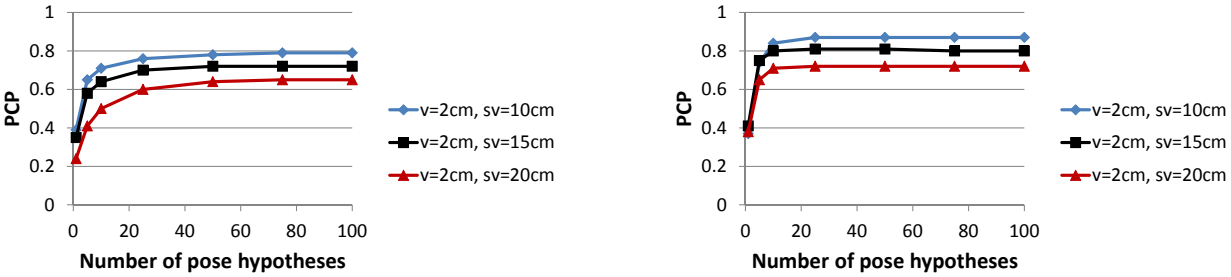
First, it will be evaluated how many pose hypotheses are required for best estimation results. Then, for the determined number of hypotheses, the impact of voxel and supervoxel sizes will be evaluated. It follows an evaluation of various connection weight thresholds for the determined best voxel and supervoxel sizes.

Number of Pose Hypotheses

The number of pose hypotheses determines the number of sampled poses during the down path in Algorithm 4, as discussed in Section 4.2.3. The more poses are sampled, the higher is the likelihood that a good pose is contained in the sample set. The following experiment determines the number of samples for which the classification scores converge.

This evaluation uses voxels with a fixed size of 2 cm and three supervoxel sizes: 10 cm, 15 cm, and 20 cm. The supervoxel graph connection weight threshold was set to 1.0. The sample size for pose hypotheses is increased from 1 to 100.

The results are shown in Figure 4.8. The PCP scores quickly increase for an increasing number of hypotheses. For more than 50 samples, the PCP scores increase very little and



(a) HumanEva-I (b) UMPM (Orthosyn sequence)

Figure 4.8: Evaluation of the number of pose hypotheses. For both datasets, good results are achieved starting with approximately 50. Smaller supervoxel sizes (sv) lead to better results. The voxel size (v) was constant.

it converge for a sample size of approximately 75. The supervoxel sizes have an effect on the overall PCP results, as will be further investigated in the next section, but they do not influence convergence behavior.

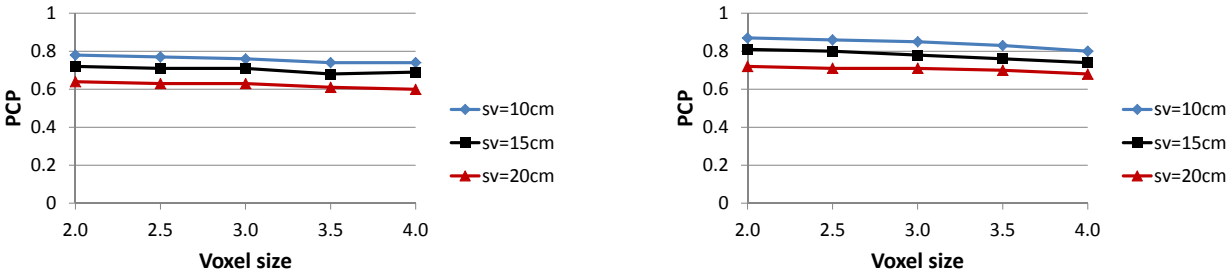
In general, a sample size between 25 and 50 is sufficient for good results. For optimal results, a sample size of 75 or greater is recommended.

Voxel and Supervoxel Sizes

The next evaluation investigates the effect of voxel and supervoxel sizes on part classification. The voxel size determines the accuracy of the 3D reconstruction. The supervoxel size directly influences the pose estimation accuracy. Joints are constrained to supervoxel centers, as explained in Section 4.1. Therefore, the larger the supervoxels, the larger can the potential deviation from the ground truth joint position be - even if the best supervoxel was chosen. For example, with supervoxel sizes of 20 cm, the distance of an optimal supervoxel to the corresponding ground truth can be up to 10 cm.

For this evaluation, the number of pose hypotheses was set to 50 and the supervoxel graph connection weight threshold to 1.0. The voxel sizes were increased in 0.5 cm increments between 2 cm and 4 cm and three supervoxel sizes were used: 10 cm, 15 cm, and 20 cm.

The results in Figure 4.9 show that voxel sizes have a low influence on the overall PCP results. The PCP results are best for the smallest voxel sizes and decrease only slightly for larger voxels. This shows that the supervoxels are suitable as joint candidates independently of voxel sizes.



(a) HumanEva-I

(b) UMPM (Orthosyn sequence)

Figure 4.9: Evaluation of voxel and supervoxel sizes. The PCP scores are best for the smallest voxel sizes and decrease for larger voxels. The same hold for supervoxels sizes and best results are achieved for the smallest supervoxels.

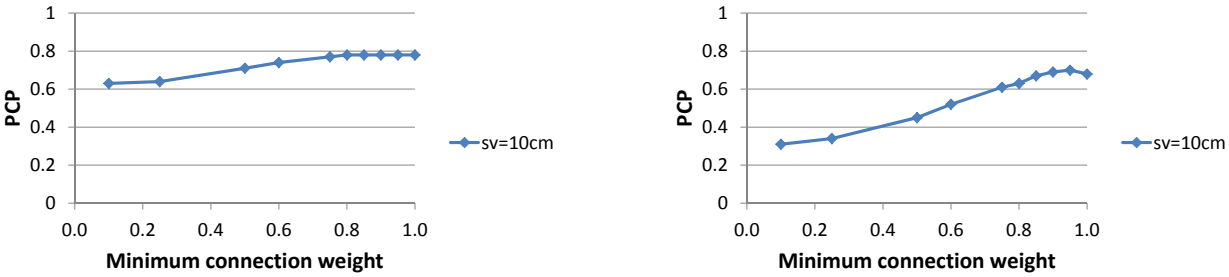
The supervoxel sizes have a stronger impact on PCP results as Figure 4.9 shows. The lower the supervoxel size, the better are the results. The reason is that the supervoxel sizes directly affect the possible deviations from ground truth joint locations as discussed above. For optimal results, the smallest supervoxels should be used. However, supervoxel sizes also determine the search space of the algorithm, as discussed in Section 4.4. Therefore, if computational efficiency is important, a larger supervoxel size can be used.

Supervoxel Graph Connection Weights

The weight of the supervoxel graph edges represents their fraction inside the volume, as explained in Section 3.3.3. For example, an edge contained inside the volumetric reconstruction by 40 % has a weight of 0.4. This threshold is intended to introduce a certain robustness against voxel carving errors. For example, if only one voxel on a graph edge is erroneously carved, *e.g.*, due to noise, the whole connection would be removed. The lower the weight threshold, the more connections does the supervoxel graph have and, consequently, the higher is the number of limb candidates.

For this evaluation, the voxel size was set to 2 cm and the supervoxel size to 10 cm. The number of pose hypotheses was 50. The connection weight threshold was varied between 0.0 and 1.0.

Figure 4.10 shows the results for this evaluation. As Figure 4.10a shows for the HumanEva-I dataset, the PCP results converge for a threshold of 0.8 or larger. The lower the threshold, the worse the results. The reason is that too many candidates are available for pose estimation, including many poor candidates that are largely outside the actual volumetric



(a) PCP scores for HumanEva-I

(b) PCP scores for UMPM (chair sequence)

Figure 4.10: Evaluation of supervoxel graph connection weights. (a) shows that relaxing the connection weight threshold is not required for the HumanEva-I dataset. (b) shows that best results are achieved with a threshold of 0.95 for the UMPM dataset sequence *Chair* with static occlusions.

reconstruction. These excess candidates negatively affect pose estimation. Also, the foreground segmentations are quite good due to the GrabCut algorithm [136] and because there are no static occlusions. Therefore, relaxing the connection weight threshold is not necessary.

As Figure 4.10b shows, this is different for the UMPM dataset with worse foreground segmentations and static occluders. Here, a slight relaxation of the connection weight to 0.95 leads to better overall results.

4.5.3 Body Tracking Evaluation

This section shows the evaluation of the body tracking algorithm with temporal pictorial structures as introduced in Section 4.3. For tracking, the limb scores of the best previous poses are propagated to the current frame. This influences pose estimation and steers poses into the direction of previously selected poses. For tracking, there is only one free parameter which is the size of the tracking history. This parameter determines the maximum time difference to the current frame for propagation of information about previous poses.

For evaluation, the voxel size was set to 2 cm and the supervoxel size to 10 cm, as it gives the best results. Varying the voxel size does not significantly affect pose estimation as long as it is below 3 cm, as shown in Section 4.5.2. The supervoxel graph connection weight threshold was set to 1 for evaluation on the HumanEva dataset and to 0.95 for the UMPM dataset. The number of pose hypotheses was set to 50. For the sole tracking parameter, *i.e.*, the history size, three settings were evaluated: 10, 20, and 30 frames.

Table 4.1: Tracking results for HumanEva-I. The table shows tracking results for three different tracking history sizes on the HumanEva-I dataset with a voxel size of 2 cm, a supervoxel size of 10 cm, and connection threshold 1.

	No tracking	History: 10	History: 20	History: 30
Head	0.96	0.97	0.97	0.97
Upper arms	0.54	0.57	0.57	0.56
Lower arms	0.27	0.29	0.29	0.29
Torso	1.00	1.00	1.00	1.00
Upper legs	0.97	0.98	0.98	0.98
Lower legs	0.93	0.94	0.94	0.94
Average	0.78	0.79	0.79	0.79

Table 4.1 shows the results of the tracking evaluation for HumanEva-I and Table 4.2 for UMPM. In general, tracking has a positive effect on the percentage of correctly classified parts. This effect is stronger for more difficult sequences with lower recognition rates. Best results are achieved for the *Table* sequence of the UMPM dataset with an PCP increase from 0.61 to 0.69. All parts benefit from tracking, *e.g.*, torso recognition rates increase from 0.92 to 0.95 for the UMPM *Chair* sequence. Results are best for tracking history sizes of 10 and 20. Overall improvements through tracking are lower for HumanEva-I. However, the most difficult parts, *i.e.*, the arms, have the highest increase by up to three percent.

4.5.4 Part Accuracy Evaluation

The previous sections showed average classification rates for all body parts. Here, the classification rates of specific parts will be discussed. For evaluation, three voxel and supervoxel sizes were used. The supervoxel graph connection weight was set to 1 for HumanEva-I and 0.95 for UMPM. The number of pose hypotheses was 50.

The results for both datasets are shown in Table 4.3. As typical for pose estimation, the classification rates decrease for limbs that are lower in the kinematic chain of the body model. The reason is that poorly estimated parent limbs directly influence their child limbs, as can be seen for upper and lower arms.

The torso, upper legs, and lower legs show high classification rates, in particular for smaller voxel and supervoxel sizes. The torso is recognized best and is classified correctly for at

Table 4.2: Tracking results for the UMPM dataset. The table shows tracking results on the UMPM dataset with a voxel size of 2 cm, a supervoxel size of 10 cm, and connection weight threshold 0.95. Two sequences with occlusions, *Chair* (C) and *Table* (Ta), are compared for three tracking history sizes to results without tracking.

	No tracking		History: 10		History: 20		History: 30	
	C	Ta	C	Ta	C	Ta	C	Ta
Head	0.68	0.55	0.72	0.62	0.73	0.64	0.72	0.64
Upper arms	0.64	0.57	0.67	0.63	0.68	0.65	0.67	0.66
Lower arms	0.42	0.40	0.44	0.45	0.43	0.46	0.43	0.47
Torso	0.92	0.88	0.95	0.94	0.95	0.98	0.96	0.97
Upper legs	0.84	0.70	0.87	0.77	0.87	0.78	0.88	0.78
Lower legs	0.69	0.58	0.71	0.61	0.71	0.62	0.71	0.62
Average	0.70	0.61	0.73	0.67	0.73	0.69	0.73	0.69

least 95 % of all evaluated sequences and sizes. The lowest scores are achieved for the upper and lower arms, in particular for the HumanEva-I dataset. There are three reasons. First, the arms are the most flexible parts and therefore harder to classify than other body parts. This is also the case in other approaches, *e.g.*, [29, 45, 134]. Second, they are relatively small. Therefore, inaccuracies introduced due to supervoxel sizes have a higher impact on their PCP scores, which are defined based on part lengths. Third, the arms are in particular poorly recognized for sequences where they are close to the body and bent, *e.g.*, in the *Box* or *Jog* sequences. This is difficult to recognize for the proposed approach because the pose prior favors extended and non-overlapping limbs. However, as the results for improved voxel data will later show, even these difficult sequences can be recognized reliably.

In the remainder of this section, the single action sequences will be discussed. It will be shown that arms are significantly better recognized for sequences with gestures, *i.e.*, where they are not too close to the body. Further, the next sections will show improved results for better voxel reconstructions (here: synthetic voxels) and by using part detectors.

The previously discussed results focused on average recognition rates over all sequences. Now, the recognition rates for single sequences will be discussed. For this evaluation, the voxel size was set to 2 cm and the supervoxel size to 10 cm. The number of pose hypotheses was 50 and the connection weight threshold 1 for HumanEva-I and 0.95 for UMPM.

Table 4.3: Part classification evaluation for the HumanEva-I (HE) and UMPM datasets. The table shows the PCP results for each body part and three voxel and supervoxel sizes. The connection threshold was 1 for HumanEva-I and 0.95 for UMPM.

	Voxel size: 2 cm SV size: 10 cm [PCP]		Voxel size: 3 cm SV size: 15 cm [PCP]		Voxel size: 4 cm SV size: 20 cm [PCP]	
	HE	UMPM	HE	UMPM	HE	UMPM
Head	0.96	0.73	0.92	0.67	0.80	0.68
Upper arms	0.55	0.70	0.41	0.53	0.24	0.39
Lower arms	0.28	0.48	0.11	0.30	0.05	0.15
Torso	1.00	0.96	0.99	0.95	0.96	0.95
Upper legs	0.98	0.88	0.95	0.82	0.86	0.80
Lower legs	0.93	0.78	0.86	0.68	0.71	0.63
Average	0.78	0.75	0.71	0.66	0.60	0.60

Table 4.4 shows results for single sequences of the HumanEva-I dataset. Except for the arms, the classification rates are generally above 90 %. When examining the classification scores for the upper and lower arms, the worst results are achieved for sequences where they are close to the body, *i.e.*, *Box* (B) and *Jog* (J). The other sequences, in particular the *Gestures* (G) and *Walking* (W) sequence, achieve significantly better results. This is important for this work which focuses on human-machine interaction. Here, arms must generally be recognized if they are extended, *e.g.*, for gesture interaction. In these cases, the presented approach achieves good results.

Similar observations can be made for the UMPM dataset results shown in Table 4.5. Here, however, the arms are recognized better, but still significantly worse than larger parts like torso and legs. For the UMPM dataset, the challenging scenarios are the ones with occlusions, like *Chair* and *Grab*, but in particular the *Table* sequence. Here, the voxel reconstructions contain too many errors that have a negative impact on pose estimation. Results for synthetic voxel data in Section 4.5.6, however, will show that without these errors, PCP scores increase up to 0.98.

Table 4.4: Classification results for single sequences of the HumanEva-I dataset.

	Actor S1					Actor S2					Average
	B	G	J	T	W	B	G	J	T	W	
Head	0.95	0.97	0.97	0.93	0.99	0.94	0.96	0.98	0.95	0.99	0.96
Upper arms	0.33	0.73	0.25	0.46	0.82	0.51	0.64	0.34	0.67	0.73	0.54
Lower arms	0.14	0.62	0.09	0.15	0.42	0.09	0.33	0.10	0.36	0.45	0.27
Torso	1.00	1.00	0.99	0.99	1.00	1.00	0.99	1.00	1.00	1.00	1.00
Upper legs	1.00	0.98	0.94	0.97	0.98	0.99	0.97	0.97	0.98	0.98	0.97
Lower legs	0.97	0.90	0.81	0.87	0.93	0.99	0.96	0.91	0.98	0.97	0.93
Average	0.73	0.87	0.67	0.73	0.86	0.75	0.81	0.72	0.82	0.85	0.78

Table 4.5: Classification results for single sequences of the UMPM dataset.

	Chair	Grab	Orthosyn	Table	Triangle	Average
Head	0.68	0.68	0.90	0.55	0.85	0.73
Upper arms	0.64	0.67	0.79	0.57	0.80	0.70
Lower arms	0.42	0.36	0.61	0.40	0.63	0.48
Torso	0.92	1.00	1.00	0.88	1.00	0.96
Upper legs	0.84	0.92	0.99	0.70	0.96	0.88
Lower legs	0.69	0.79	0.97	0.58	0.87	0.78
Average	0.70	0.73	0.88	0.61	0.85	0.75

Table 4.6: Part detector evaluation on the HumanEva-I (HE) and UMPM datasets. This table shows the classification improvements when using (simulated) part detectors.

Body part	Voxel size: 2 cm SV size: 10 cm [PCP]		Voxel size: 3 cm SV size: 15 cm [PCP]		Voxel size: 4 cm SV size: 20 cm [PCP]	
	HE	UMPM	HE	UMPM	HE	UMPM
Head	1.00	0.98	0.99	0.96	0.98	0.93
Upper arms	0.92	0.93	0.71	0.80	0.51	0.62
Lower arms	0.67	0.87	0.36	0.67	0.20	0.39
Torso	1.00	1.00	1.00	1.00	1.00	1.00
Upper legs	0.97	1.00	0.95	0.98	0.91	0.96
Lower legs	0.95	0.98	0.89	0.95	0.81	0.91
Average	0.92	0.96	0.82	0.89	0.74	0.80

4.5.5 Part Detector Evaluation

One advantage of the pose estimation approach presented here is that it requires very little prior information. In particular, it does not need training data for part appearances, as is the case for part-detector based approaches. However, part detectors can be very useful in certain situations. As Section 4.3.3 has shown, such information can be directly integrated into the pose estimation algorithm by using supervoxel energies. This will now be evaluated.

In this evaluation, part detectors are simulated to measure their impact on pose estimation. Therefore, for each joint, energies of supervoxels are set accordingly to their distance to the corresponding ground truth joint. If a supervoxel is within a radius given by the limb size, which is twice the size used in the PCP measure, it is directly assigned its distance as energy. If it is farther away, its energy is set to infinity.

For evaluation, three voxel and supervoxel size combinations were used. The number of pose hypotheses was set to 50 and the connection weight threshold of the supervoxel graph to 1 for HumanEva-I and 0.95 for UMPM.

Table 4.6 shows that using additional information through part detectors has the potential to significantly improve the percentage of correctly classified parts for both datasets. This is in particular the case for upper and lower arms as a comparison with Table 4.3 shows.

Table 4.7: Pose estimation with synthetic voxel data on the HumanEva-I (HE) and UMPM datasets. The table shows the accuracy of the pose estimation algorithm when no voxel carving errors are present.

Body part	Voxel size: 2 cm SV size: 10 cm [PCP]		Voxel size: 3 cm SV size: 15 cm [PCP]		Voxel size: 4 cm SV size: 20 cm [PCP]	
	HE	UMPM	HE	UMPM	HE	UMPM
Head	1.00	1.00	1.00	0.98	0.97	0.97
Upper arms	0.96	0.99	0.87	0.95	0.76	0.85
Lower arms	0.78	0.94	0.64	0.79	0.49	0.64
Torso	1.00	1.00	1.00	1.00	1.00	1.00
Upper legs	1.00	0.99	1.00	0.99	0.98	0.98
Lower legs	1.00	0.98	0.99	0.96	0.97	0.93
Average	0.96	0.98	0.92	0.95	0.86	0.89

The average PCP scores for the smallest voxel and supervoxel size is 0.92 for HumanEva-I and 0.96 for UMPM.

In this work, one question is how to estimate a human pose with a minimal amount of prior knowledge. The results in the previous sections showed that one solution is by using supervoxel segmentation as a preprocessing step. With the results in this section, it was shown that through additional information, *e.g.*, by using trained part detectors, the overall recognition rates can further be improved. This is a potential direction for future work.

4.5.6 Synthetic Voxels

The pose estimation presented here works with voxels as the smallest unit of operation. Any errors introduced through voxel carving have an impact on pose estimation. As the presented evaluations with two challenging datasets showed, the algorithm is robust and works well with video data.

However, in the following evaluation, the potential maximal performance of the presented algorithm is evaluated that would be achievable if there were no errors and inaccuracies through voxel carving. Based on the ground truth body parts, synthetic voxels were

generated. This has been done in the same manner as for the supervoxel evaluation in Section 3.3.4.

For this evaluation, three voxel sizes were used in combination with three supervoxel sizes. The supervoxel graph connection threshold was set to 1 because there are no voxel carving errors for synthetic data. The number of pose hypotheses was set to 50.

Table 4.7 shows that the accuracy for the pose estimation algorithm is very high for both datasets if there are no voxel carving errors. For small voxel and supervoxel sizes, the average PCP score is 0.96 for HumanEva-I and 0.98 for UMPM and decreases with increasing voxel and supervoxel sizes. These results show that the pose estimation algorithm is able to accurately estimate human poses.

4.5.7 Evaluation of the 3D Joint Localization Error

The 3D joint localization error is the average distance between estimated and ground truth joint positions in 3D space. It is well suited for evaluation because it allows a direct comparison between different approaches based on their accuracy. Following [147, 148], the 3D error is measured in millimeters.

The parameters were chosen according to Section 4.5.4: The connection weight of the supervoxel connection graph was set to 1 for the HumanEva-I and to 0.95 for UMPM dataset. The number of pose hypotheses was 50. The voxel sizes were 2 cm, 3 cm, and 4 cm with supervoxel sizes of 10 cm, 15 cm, and 20 cm. The results for both datasets are shown in Table 4.8. Table 4.9 shows detailed results for all single sequences of the HumanEva-I dataset and Table 4.10 for the UMPM dataset.

The evaluation results are in accordance with Section 4.5.4. The 3D error is lowest for smaller voxel and supervoxel sizes and increases with larger voxels and supervoxels (Table 4.8). The results on the UMPM dataset are, on average, slightly worse than on the HumanEva-I dataset (Table 4.8). However, as Table 4.10 shows this is due to the *Table* sequences, the most challenging sequence in the evaluation. The 3D errors of the other sequences are comparable to the HumanEva-I results.

The accuracy for joints near the torso is better than for joints that are closer to the end of the kinematic chains. In particular the wrist joints are difficult to estimate. The accuracy for the other parts is significantly better. This is the case for both datasets as Tables 4.9 and 4.10 show.

Table 4.8: 3D joint localization error for the HumanEva-I (HE) and UMPM datasets.

Joint	Voxel size: 2 cm SV size: 10 cm [mm]		Voxel size: 3 cm SV size: 15 cm [mm]		Voxel size: 4 cm SV size: 20 cm [mm]	
	HE	UMPM	HE	UMPM	HE	UMPM
Head	92	137	116	157	194	158
Neck	59	65	63	76	84	74
Shoulder	82	95	88	111	124	116
Elbow	148	162	179	216	214	252
Wrist	272	226	340	315	376	368
Pelvis	59	65	63	76	84	74
Hip	71	94	74	105	98	108
Knee	81	144	97	164	145	178
Ankle	106	161	144	190	216	213
Average	115	135	139	167	181	185

The results presented here will be further discussed in Section 4.5.10 with a comparison to other approaches.

4.5.8 Qualitative Results

This section shows qualitative results of the presented pose estimation algorithm. First, images are shown for the successive steps of the algorithm starting with voxel carving and supervoxel segmentation to the final pose estimate. Figure 4.11 shows examples for the HumanEva-I dataset and Figure 4.12 for the UMPM dataset. Then, examples of pose estimates are shown for various voxel and supervoxel sizes. Results for the HumanEva-I dataset are shown in Figure 4.13 and for the UMPM dataset in Figure 4.14. In addition, images are shown for pose estimation with a single depth-sensor in Figure 4.15.

4.5.9 Runtime Evaluation

This section presents the runtime evaluation of the pose estimation algorithm. Runtime measurements are given for both CPU as well as GPU implementation. In addition to the

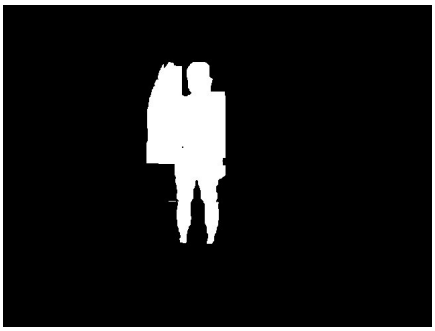
Table 4.9: 3D joint localization error for single sequences of the HumanEva-I dataset. The voxel size was 2 cm and the supervoxel size 10 cm.

	Actor S1					Actor S2					Average
	B	G	J	T	W	B	G	J	T	W	
Head	93	84	89	100	74	100	104	83	105	88	92
Neck	58	56	58	53	58	63	61	58	63	63	59
Shoulder	87	76	78	76	74	97	80	82	83	82	82
Elbow	204	122	191	158	87	164	127	191	129	109	148
Wrist	480	207	264	337	167	417	241	255	200	155	272
Pelvis	58	56	58	53	58	63	61	58	63	63	59
Hip	68	69	70	68	69	76	73	70	75	74	71
Knee	76	76	97	88	71	83	80	78	80	82	81
Ankle	87	107	155	134	101	96	101	102	89	91	106
Average	147	101	128	129	89	140	109	117	103	93	115

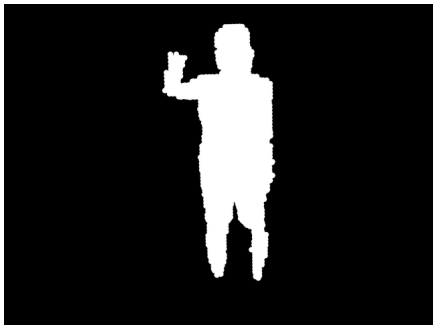
Table 4.10: 3D joint localization error for single sequences of the UMPM dataset. The voxel size was 2 cm and the supervoxel size 10 cm.

	Chair	Grab	Orthosyn	Table	Triangle	Average
Head	151	125	91	222	97	137
Neck	83	48	40	111	45	65
Shoulder	109	80	67	141	77	95
Elbow	169	161	152	201	127	162
Wrist	247	242	208	277	156	226
Pelvis	83	48	40	111	45	65
Hip	111	80	63	134	84	94
Knee	154	125	88	239	112	144
Ankle	170	132	98	293	114	161
Average	149	124	101	201	102	135

(a) Camera image

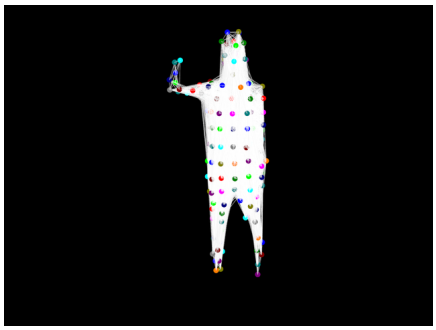


(b) Foreground segmentation

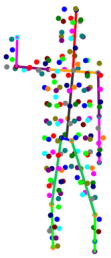


(c) Voxel carving

(d) Supervoxel segmentation

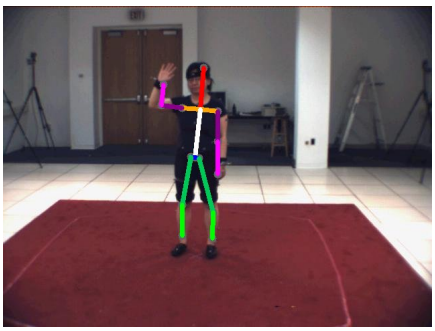


(e) Supervoxel graph



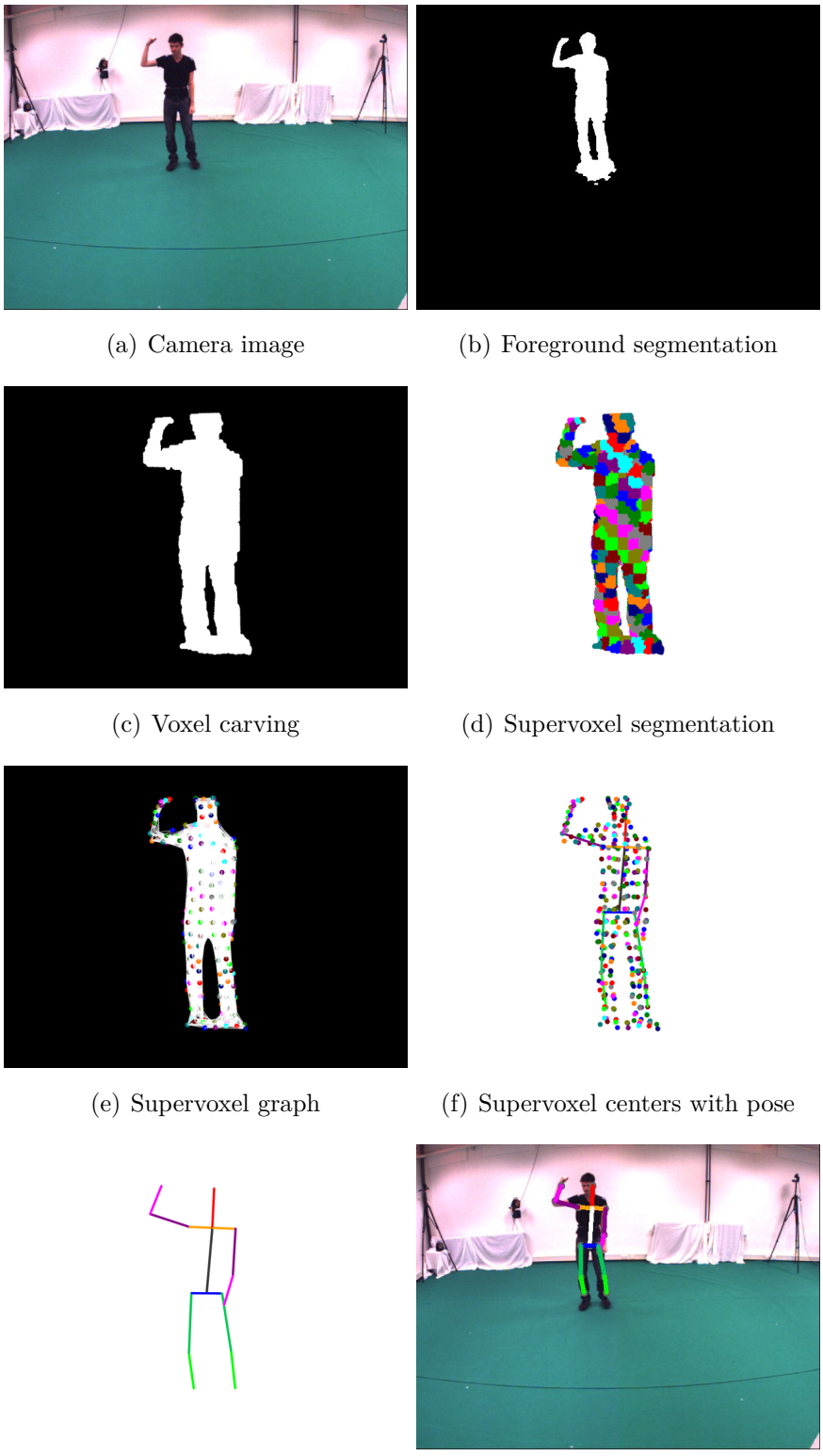
(f) Supervoxel centers with pose

(g) Result



(h) Pose projected into image

Figure 4.11: Pose estimation process on HumanEva-I dataset. The images show all steps of the pose estimation process.



(a) Camera image (b) Foreground segmentation

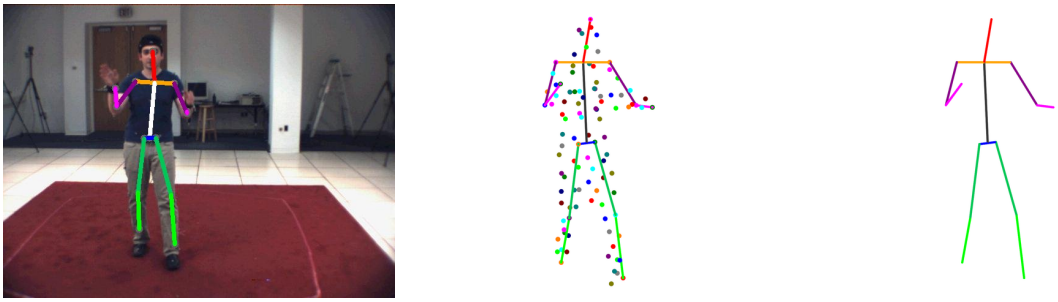
(c) Voxel carving (d) Supervoxel segmentation

(e) Supervoxel graph (f) Supervoxel centers with pose

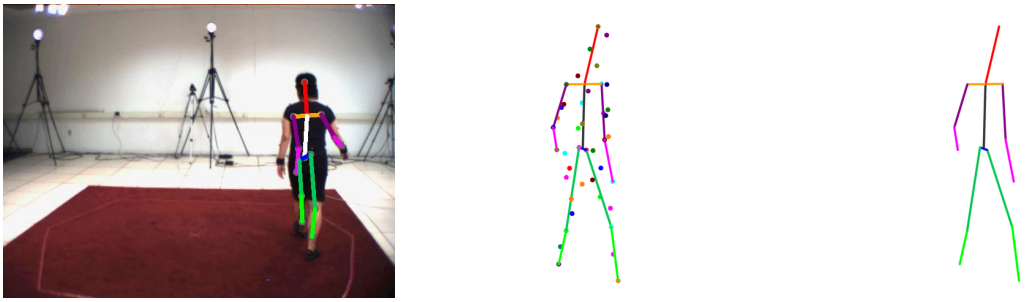
(g) Result (h) Pose projected into image

Figure 4.12: Pose estimation process on UMPM dataset. The images show all steps of the pose estimation process.

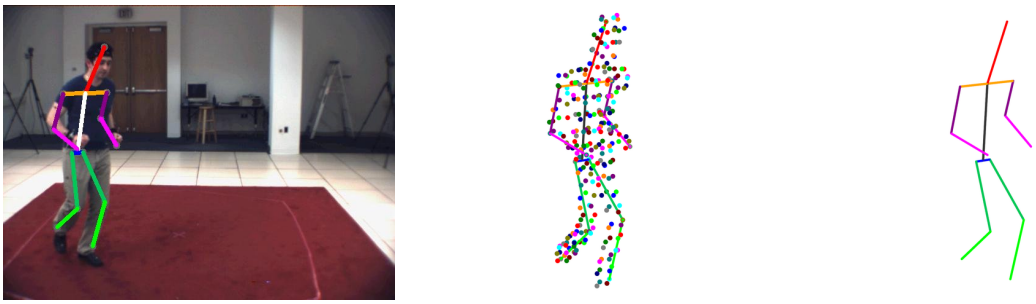
(a) S1 ThrowCatch, voxel size: 2 cm, supervoxel size: 10 cm



(b) S2 Gestures, voxel size: 3 cm, supervoxel size: 15 cm

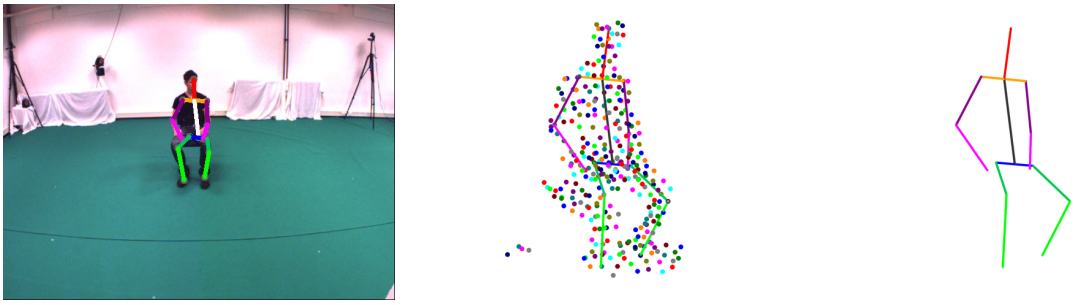


(c) S1 Walking, voxel size: 4 cm, supervoxel size: 20 cm

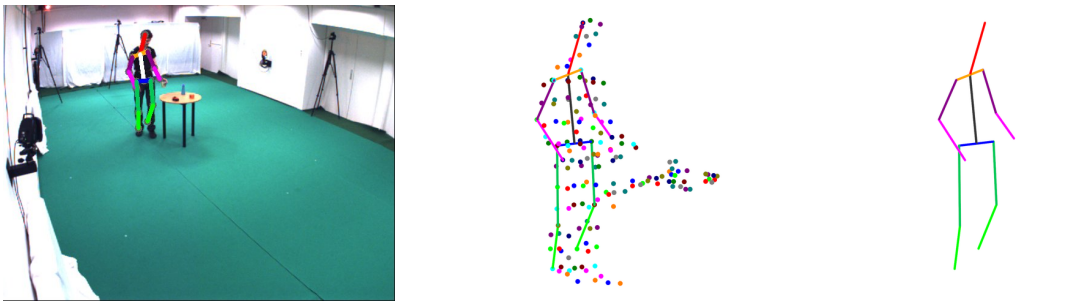


(d) S2 Jog, voxel size: 2 cm, supervoxel size: 10 cm, synthetic voxels

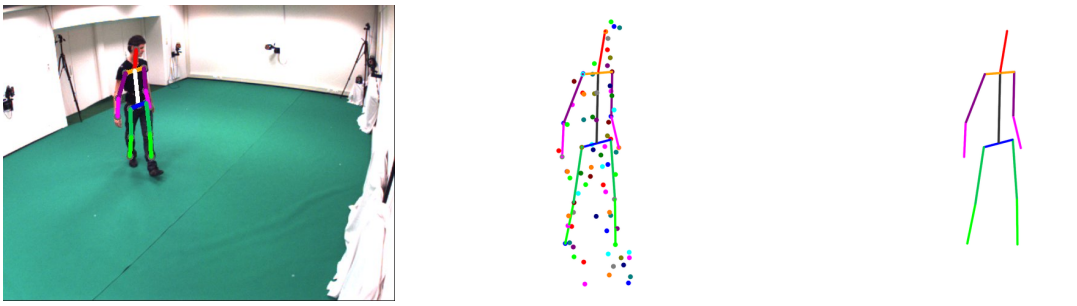
Figure 4.13: Qualitative pose estimation results on HumanEva-I dataset for various voxel and supervoxel sizes.



(a) Chair, voxel size: 2 cm, supervoxel size: 10 cm



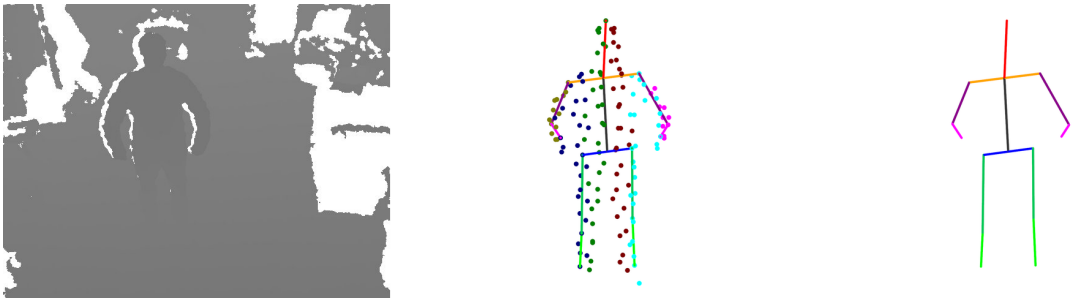
(b) Grab, voxel size: 3 cm, supervoxel size: 15 cm



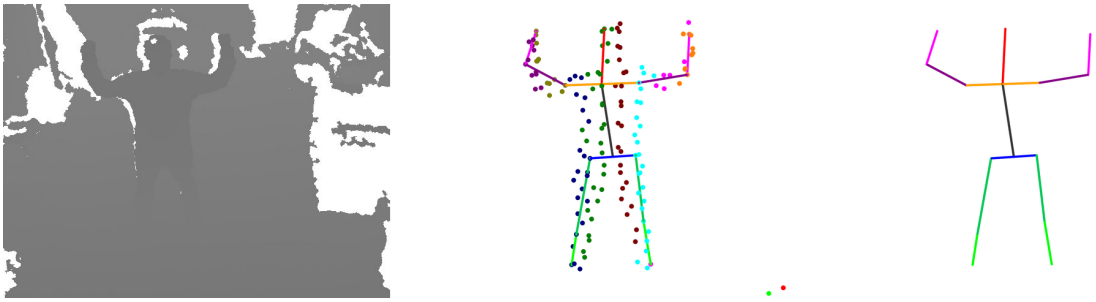
(c) Triangle, voxel size: 4 cm, supervoxel size: 20 cm

(d) Table, voxel size: 2 cm, supervoxel size: 10 cm, synthetic voxels

Figure 4.14: Qualitative pose estimation results on UMPM dataset for various voxel and supervoxel sizes.



(a) One depth sensor, voxel size: 2 cm, supervoxel size: 10 cm



(b) One depth sensor, voxel size: 2 cm, supervoxel size: 10 cm

Figure 4.15: Qualitative pose estimation results with a single depth sensor.

Table 4.11: Pose estimation part numbers. This table shows the numbers of candidates for joints, *i.e.*, supervoxels, and limbs, *i.e.*, supervoxel graph connections, for three supervoxel sizes.

	SV size: 10 cm [*Number*]	SV size: 15 cm [*Number*]	SV size: 20 cm [*Number*]
Joints candidates	182	77	46
Limbs candidates	20,556	3,382	1,358

overall runtime, the runtimes for up and down paths of Algorithm 4 and for sampling the body parts, *i.e.*, limbs, are analyzed separately. The voxel size was fixed as it does not have a direct influence on pose estimation runtimes. It was set to 2 cm because this achieved the best results. The three supervoxel sizes were 10 cm, 15 cm, and 20 cm. The supervoxel graph connection weight threshold was set to 1 and the number of pose hypotheses was 50. The measurements were taken for the first sequence of the HumanEva-I dataset, *i.e.*, *Box* with subject S1.

The main factor for runtime is the numbers of supervoxels because they determine the number of parts and connections used for pose estimation. The numbers of parts and connections for three different supervoxels sizes are shown in Table 4.11. The runtimes for pose estimation are shown in Table 4.12, again for three different supervoxel sizes. This table also includes runtimes for body tracking.

As expected, the runtimes increase with an increasing number of supervoxels. Here, both CPU and GPU implementation are significantly below 100 ms for all sizes. However, the GPU implementation does not lead to an improved runtime. The reason is that due to the search space reduction through supervoxels, the actual number of parts is not sufficiently high to fully use the power of parallel processing. One exception is a supervoxel size of 10 cm where the GPU is faster. However, the saved time gets consumed by copying data to and from the GPU. This overhead is not required for the CPU implementation.

The runtimes in Table 4.12 for the down path are for sampling 50 poses. This shows how efficient this particular step is. As a result, sampling a larger number of poses for better results would only lead a small increase in computation time.

Interestingly, in contrast to the worst case complexity discussed in Section 4.4, the overall runtimes seem to be linear in the number of supervoxels. This can be seen when comparing supervoxel numbers in Table 4.11 with runtimes in Table 4.12. Approximately half the

Table 4.12: Pose estimation runtimes. The table shows runtimes for both CPU and GPU implementations of the pose estimation algorithm. The runtimes are given for the single parts of the algorithm as well as the total time.

	SV size: 10 cm		SV size: 15 cm		SV size: 20 cm	
	CPU	GPU	CPU	GPU	CPU	GPU
Algorithm part	$[ms]$	$[ms]$	$[ms]$	$[ms]$	$[ms]$	$[ms]$
Limb candidates	17	13	1	2	1	1
Up path	37	26	2	2	< 1	1
Down path (50 poses)	16	17	4	5	1	3
GPU overhead	–	34	–	4	–	1
Body tracking	< 1	< 1	< 1	< 1	< 1	< 1
Total	71	91	8	14	4	7

number of supervoxels leads to half the computation time. The reason is that, in contrast to the worst case complexity, not all limbs are connected in the supervoxel graph.

Even with the additional computation time required for voxel and supervoxel segmentation, as discussed in Sections 3.1.4 and 3.2.3, the system runs in real-time with a supervoxel size of 15 cm or larger.

4.5.10 Comparison to the State-of-the-Art

This section compares the results of the presented pose estimation algorithm with state-of-the-art approaches. First, a comparison based on the PCP measure is presented followed by a comparison based on the 3D joint localization error.

Comparison based on Percentage of Correct Parts

The pose estimation algorithm is closest to Burenius *et al.* [45], proposed in 2013, which is one of the few 3D pose estimation approaches with pictorial structures. Their approach achieves average PCP scores of up to 0.77. However, a direct comparison of PCP results is difficult because only selected frames of a special dataset have been evaluated in [45]. But as one of their main contributions was a reduction of the overall complexity, the runtimes can be compared. (The complexity was already compared in Section 4.4.) For

their recommended discretization sizes of translation space ($|T| = 32^3$) and rotation space ($|R| = 8^3 \vee 16^3$), Burenius *et al.* [45] reported runtimes that are between 1 second and 69 minutes. In contrast, even with the lowest voxel and supervoxel resolutions, the runtimes presented here are below 100 milliseconds.

In 2013, Ramakrishna *et al.* [134] reported results on the HumanEva-I dataset using the PCP measure. Their tracking approach explicitly uses the symmetric structure of human bodies and works on single images by using part detectors. They reported results for the first 250 frames of the following sequences: S1 Walking, S1 Jog, and S2 Jog. Their reported PCP results are between 0.98 and 1.00 for all parts except lower arms which achieve a PCP score of 0.53. Their results are better than the results reported in this work, in particular for the arms. However, with (simulated) part detectors, the differences diminish and a comparable performance is achieved (Table 4.6). A comparison of runtimes is not possible as they were not reported.

Comparison based on 3D Joint Localization Error

This section compares the presented pose estimation approach with other approaches. The presented pose estimation approach achieves a 3D joint localization error of 115 mm with voxel sizes of 20 mm and supervoxel sizes of 100 mm on the HumanEva-I dataset. Detailed results were presented in Section 4.5.7. The comparison to other approaches focuses mostly on the HumanEva datasets because there are currently no published results for comparable approaches on the UMPM dataset. However, given the more challenging sequences of the UMPM dataset, a 3D error of 135 mm is comparable to the results on the HumanEva-I dataset.

Canton-Ferrer *et al.* [48] evaluated their tracking approach based on an annealed particle filter on the HumanEva-I dataset. Similar to the presented approach, they also used a voxel reconstruction as input with voxel sizes of 20 mm. As discussed in Section 2.3.3, their approach is computationally expensive as a large number of hypotheses must be tracked and evaluated simultaneously. The reported 3D joint localization error 121.18 mm.

Cheng and Trivedi [51] also used a voxel representation. They applied Gaussian mixture models to represent body parts and voxel data as input. This system requires a precise initialization and is computationally complex (see Section 2.3.3). It achieved a 3D error of 159 mm on the HumanEva-II dataset.

Amin *et al.* [26] infer the 3D pose from multiple 2D poses that were estimated based on a pictorial structures approach. They evaluated their algorithm on different subsets of the

HumanEva dataset and report 3D errors between 47.7 mm and 62.4 mm. This approach achieves lower 3D errors than the presented work, but relies on trained appearance-based part detectors for 2D pose estimation. The runtime has not been evaluated.

Kanaujia *et al.* [86] trained a part-based detector directly on voxel data. As one of their contributions, they applied the basic idea used in the Microsoft Kinect [146] to voxels. After adjusting the orientation of the visual hull, the system classifies voxels with a Support Vector Machine trained with 3D Shape Context Histogram features. The system was evaluated on various sequences of the HumanEva dataset and the reported 3D error is in the range of 71.261 mm and 90.952 mm. The runtime of the system has not been reported.

4.6 Conclusion

This concludes the pose estimation and body tracking section. An approach was presented that uses supervoxels as building blocks for frame-based pose estimation. By restricting joints to supervoxel centers and limbs to connections of the supervoxel graph, the complexity as well as search space was significantly reduced. Further, the approach requires little prior knowledge and does not rely on large training datasets. In addition, extensions to articulated body tracking and part detectors were introduced that can be directly integrated into the pose estimation system through the concept of supervoxel energies.

The evaluation on two datasets showed that the presented approach can estimate the articulated poses of humans. For the smallest supervoxel size of 10 cm, the average PCP scores for the HumanEva-I dataset are 0.78 and 0.75 for the UMPM dataset with 3D joint localization errors of 11.5 cm and 13.5 cm, respectively. Results are further improved with body tracking by up to eight percent for sequences of the UMPM dataset. Evaluations with simulated part detectors and with synthetic data showed that the average performance can be further increased to 0.92 and 0.96, respectively. Due to the computational complexity and search space reduction through supervoxels, the runtimes are in the range of 71 ms for supervoxel sizes of 10 cm and down to 4 ms for supervoxel sizes of 20 cm, thus enabling real-time interaction.

5 Applications

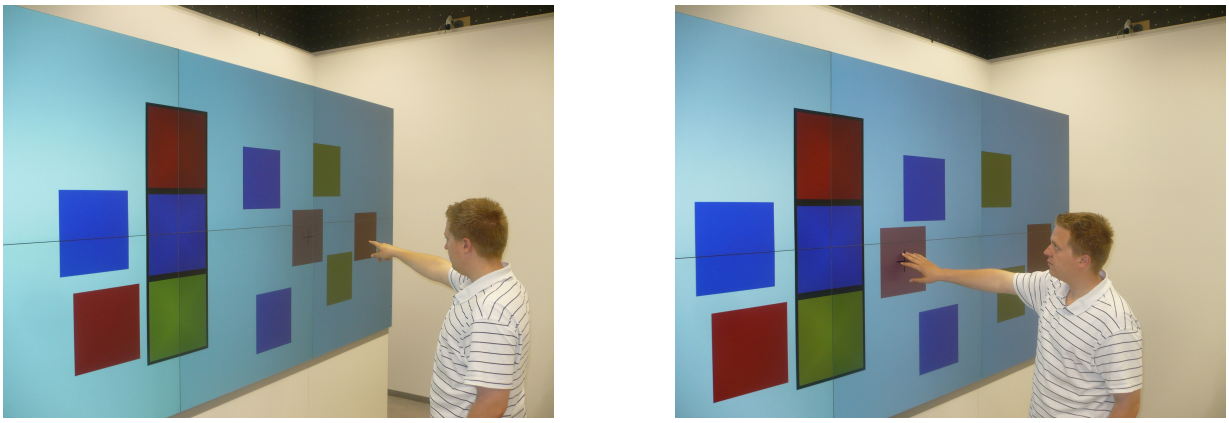
This work was developed in the context of human-machine interaction in smart environments [10, 169]. All algorithms described in the previous sections were combined to a system that allows natural and intuitive interaction in and with these smart environments. The main challenge was to develop a system that allows for real-time interaction with a robustness that not only allows applications in laboratory settings, but also in real-world industrial environments.

The next sections show three example applications that recognize both static and dynamic gestures based on the algorithms developed in this work. While static gestures are the result of classifying a specific static pose, dynamic gestures also include temporal information through following the arm trajectories.

Section 5.1 first introduces the basic gesture recognition system that is also the basis for all following applications. This system recognizes static gestures, in particular pointing. It was developed for interaction with large displays in the SmartControlRoom [10] at the Fraunhofer Institute of Optronics, System Technologies and Image Exploitation IOSB in Karlsruhe. Then, Section 5.2 shows how this system was enhanced to also recognize arbitrary dynamic gestures which is demonstrated with mid-air handwriting recognition. The third application described in Section 5.3 uses pointing gesture recognition in an industrial setting. Here, workers are enabled to directly interact with real-world objects through gestures for an improved quality assurance process.

5.1 Pointing Interaction with Large Videowalls

This section describes the pointing gesture recognition system. It was the first application developed in the context of this work and is the basis for all following applications. The main challenge was to build a real-time system that allows both touch and pointing interaction with large videowalls. Figure 5.1 shows images of both modalities. This system was first published in [8] with an additional investigation of specific interaction techniques in [1].



(a) Pointing interaction

(b) Touch interaction

Figure 5.1: Touch and pointing interaction. The left image shows the selection of a distant object through pointing. The right image shows how interaction can seamlessly be continued with direct touch. The images were first published in [8].

The system was also displayed during the SmartControlRoom demonstration at the *CeBIT 2011*, as shown in Figure 5.2, which was one of the top-10 exhibitions.

5.1.1 Arm Segmentation

The gesture recognition system directly works on voxels resulting from the voxel carving algorithm introduced in Section 3.1. In fact, with limiting assumptions about the environment, pointing gesture recognition does not necessarily require the whole body pose. However, given a pose estimate, gesture recognition can be further improved. First, the original system will be introduced. Then, the improvement through pose estimation will be shown.

The system consists of two parts. The first part segments and recognizes extended arms that are pointing towards the videowall. The second part tracks the arm over time and recognizes *clicking* events, similar to mouse clicks, that can be used as input for applications.

The arms are recognized by clustering and analyzing the voxels belonging to the visual hull. Let $\mathfrak{V}$ be the voxel grid after carving with remaining voxels $v \in \mathfrak{V}$ with side length c.

(a) (b)

Figure 5.2: Pointing interaction at CeBIT 2013. The images show the pointing gesture system as part of the SmartControlRoom demonstration.

Further, let $d(v, w)$ give the Euclidean distance between voxel centers of v and w. Then, there exists a path $p(v, w)$ between two voxels if the following condition is met:

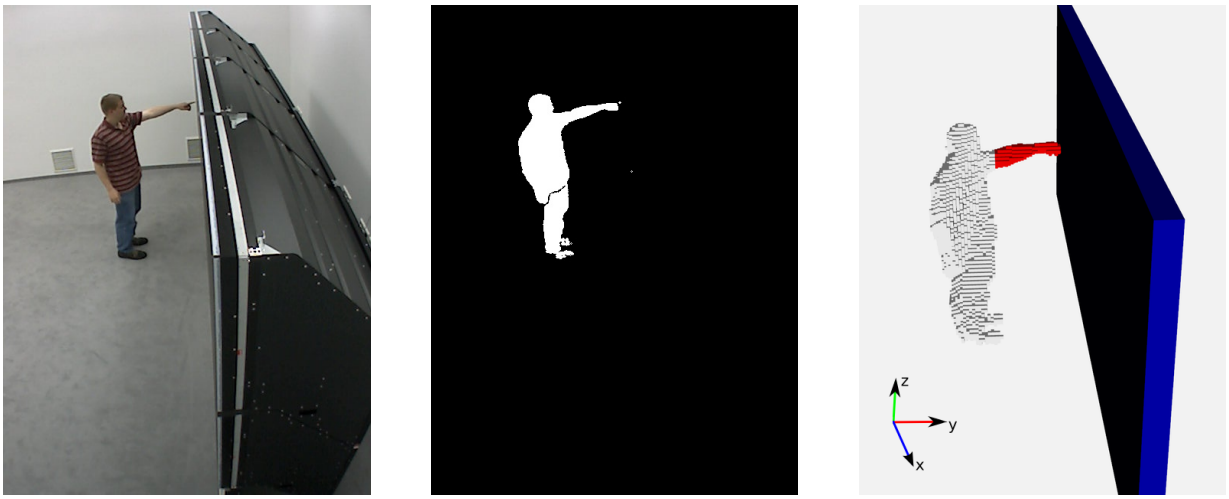
$$p(v, w) \iff d(v, w) = c \vee \exists u \in \mathfrak{V} : p(v, u) \wedge p(u, w). \tag{5.1}$$

Based on the definition of a path, voxel clusters C_v are defined as groups of voxels where a path exists between all voxels. These clusters are initialized with voxels that are closest to the videowall. Their maximum length along the orthogonal axis of the videowall is restricted to the size c_{max}. Then, a cluster with seed voxel v is defined by

$$C_v = \{w \in \mathfrak{V} : p(v, w) \wedge d(v, w) < c_{max}\}. \tag{5.2}$$

This clustering procedure requires that the direction of interaction is known, which in this case is the videowall. Further, only arm-shaped clusters are valid candidates for pointing gestures. Therefore, clusters with a higher width than length are filtered. The remaining clusters are then treated as arm candidates. Figure 5.3 shows an example camera image with foreground segmentation and a 3D view of the voxels with the segmented arm.

Based on the extracted and filtered clusters, the pointing direction is determined. Because of the assumption that gestures are pointing towards the videowall, the 3D vector representing the pointing direction can be computed with a double linear regression along the axis



(a) Camera view (b) Foreground segmentation (c) Voxels with segmented arm

Figure 5.3: Arm detection based on 3D reconstruction. Using the camera view and foreground segmentation, the 3D reconstruction is computed with voxel carving. Through clustering the voxels, the pointing arm is segmented. The images were first published in [8].

orthogonal to the videowall. With the assumption that the y-axis is orthogonal to the videowall, the 3D pointing vector l is given by

$$l_v = \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} b_1 \\ 0 \\ b_2 \end{pmatrix} + \begin{pmatrix} a_1 \\ 1 \\ a_2 \end{pmatrix} y \tag{5.3}$$

and parameters a_1, a_2, b_1, b_2 can be estimated through linear regression with voxels of cluster C_v. The pointing location in display coordinates is then given by intersecting vector l with the known 3D plane of the display.

The arm segmentation presented above works in restricted environments where assumptions about the direction of interaction can be made. Given a pose estimate, as described in Section 4, the clustering can be improved by directly assigning voxels to the closest body parts. Figure 5.4 shows example images for pose-based voxel clustering. The advantage is not only a more robust segmentation process, but also fewer assumptions. Here, arbitrary pointing directions are possible without limiting the direction of interaction, as is required above.

Figure 5.4: Clustering voxels based on pose estimation. The images show examples were voxels are clustered with respect to the closest body part. The different colors represent the clusters. Arbitrary pointing directions can be recognized which would not be possible with the pure voxel-based clustering approach.

5.1.2 Tracking

The system described above segments arms in voxel data and determines the pointing direction and location on the display. Generally, applications not only require a location, but also an interaction event. In the simplest form, this is a *clicking* event. As the example of the mouse shows, such a simple event already allows quite complex applications.

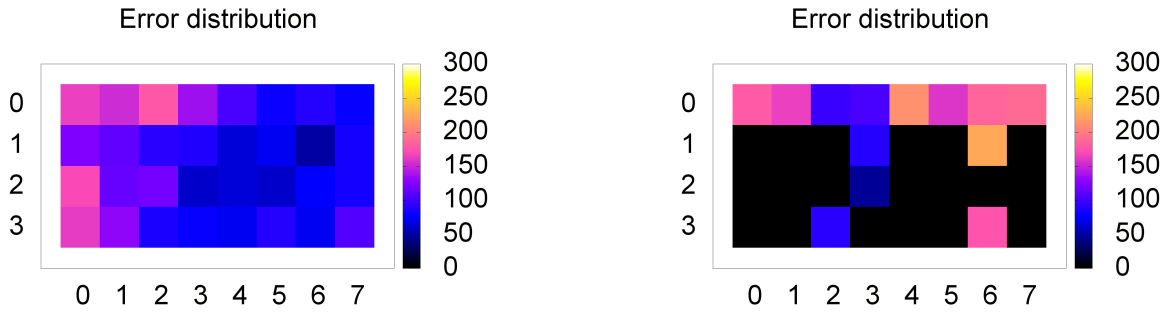
In this work, a clicking event is triggered if the arm remains still for a certain time threshold, also called dwell time. Other clicking modalities were also investigated and published in [9], but are not relevant for this work. Detecting when an arm remains still, requires tracking over time.

In this application, tracking is done by assigning clusters C^t of the current frame t to the closest cluster from previous time step $t-1$. A clicking event is then triggered if the arm remains still, *i.e.*, its movement is below a certain distance threshold, for the specified dwell time. For real-world applications, the dwell times were chosen to be between 0.5 and 1 seconds.

5.1.3 Evaluation

The pointing gesture recognition system was evaluated both from a technical viewpoint as well as in the context of a user study. Here, only the accuracy results are presented. Additional evaluations can be found in [1, 8].

All experiments were done in the SmartControlRoom at the Fraunhofer IOSB. The experimental setup included a videowall for interaction with a screen resolution of 4096×1536



(a) Touch accuracy

(b) Pointing accuracy

Figure 5.5: Accuracy evaluation of touch and pointing gestures. The left image shows the error distribution for touch and the right image for pointing interaction across the videowall. The error values are measured in pixels. Black areas indicate that no interaction took place. These results and images were first published in [8].

pixels and a display size of 4×1.5 m with the highest point at 2.3 m. Two RGB network cameras mounted to the walls next to the videowall with a resolution of 640×480 pixels were used as input.

For the experiment to evaluate the accuracy of the pointing recognition, five users were asked to point at or touch 128 evenly distributed targets on the videowall. The participants did not receive any feedback at all during this experiment. Based on their gestures, the mean error and standard deviation of the estimated and actual pointing location were computed. Figure 5.5 shows the error distributions across the videowall. As expected, touch is more accurate than pointing. One reason is that with increasing distance, even small arm movements lead to an increasingly larger offset of the pointing direction. Further, users did not receive any feedback at all. However, as following work showed [1], users can interact with objects down to 25 pixels side length, which is approximately 2.5 cm, if they receive feedback, *e.g.*, in form of a cursor.

This concludes the introduction of the touch and pointing gesture recognition system developed in this work. It allows for real-time human-machine interaction and can both be used directly with voxels or based on pose estimation results. The next sections shows how this base system can be applied to two additional applications.

Figure 5.6: Handwriting recognition in mid-air. The system recognizes characters and words that are written in front of the videowall. This image was published in [5].

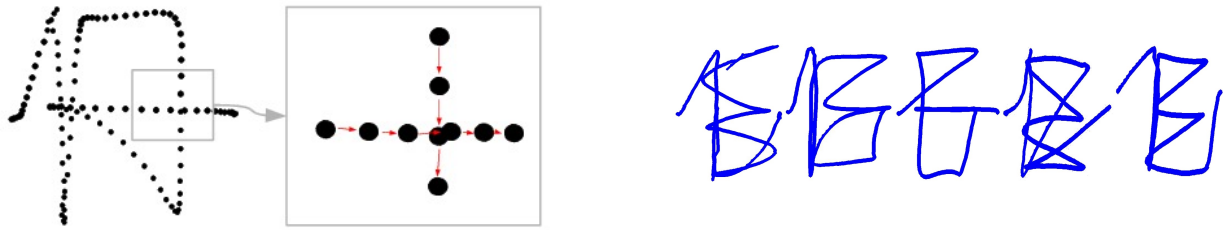
5.2 Handwriting Recognition in Mid-Air

The previous section described the recognition of static gestures for pointing interaction. The challenge addressed in this section is the recognition of arbitrary dynamic gestures based on arm movements. This is demonstrated with an example application that allows writing characters and words in mid-air. The system was developed during the diploma thesis of Daniel Morlock [15] and published in [5]. An example of the system is shown in Figure 5.6. Basis and motivation for this application was the wearable air writing system presented by Amma *et al.* [27].

Handwriting recognition is well suited as a use case for general gesture recognition. The single characters represent arbitrary dynamic gesture symbols and words are concatenations of such symbols. The recognition follows established handwriting recognition approaches [126] and uses Hidden Markov Models (HMM) [133] for both character as well as word recognition.

The system starts gesture recognition as soon as an extended arm is recognized by the system described in Section 5.1. It then extracts features from the arm trajectory that are used for character and word recognition. The gesture recognition stops if the arm is moved to a resting position.

The features used for recognition are the connecting vectors between sampled points of the hand trajectory that were projected onto the videowall surface. The videowall and, in particular, the feedback displayed on the videowall is not necessary and the system can also work with virtual planes in front of the users. The extracted features are shown in Figure 5.7 as well as varying trajectories for one example letter. The left-to-right HMMs



(a) Feature extraction from arm trajectory

(b) Writing style variations

Figure 5.7: Features and trajectories for handwriting recognition in mid-air. (a) shows the features represented by connecting vectors between points sampled from the hand trajectories. (b) shows an example of the variations when different people write the same character, here *E*. The images were also published in [5].

were trained in a person-independent way. The training dataset for characters contained all characters of the English alphabet written by five people for five times, resulting in a total of 25 samples for each character. For word recognition, 40 words were written by 16 users for one time. The HMMs for word recognition were created by concatenating the single character HMMs with an additional refinement step based on the recorded word samples.

The evaluation of the character recognition system followed a leave-one-out cross-validation scheme and achieved recognition rates of up to 86.15 %. The confusion matrix for character recognition can be seen in Figure 5.8. The word recognition evaluation also followed a leave-one-out cross-validation scheme and achieved recognition rates of 97.54 %. It can be assumed that the recognition rate would be lower for a larger number of words. However, as an example for general dynamic gesture recognition, 40 words using 26 characters are sufficient to cover most real-world use cases.

This section explained how dynamic gestures can be recognized based on the system introduced in Section 5.1. The task of recognizing characters and words based on arm movements is quite complex. There is a large variability how different users write each character. Further, there is a large number of possible character combinations. By demonstrating that characters and words can be reliably recognized, it was shown that the system is suitable for general gesture recognition, which concludes this section.

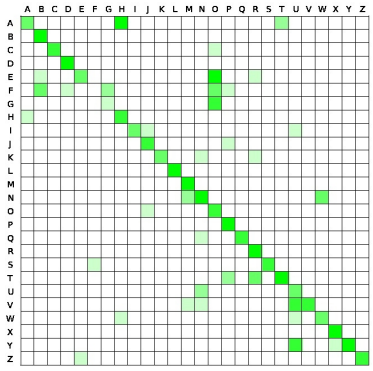


Figure 5.8: Confusion matrix for handwriting recognition. The matrix shows that most characters are recognized reliably. The confusion between F and G is due to the continuous writing style. This image was also published in [5].

5.3 Gesture Recognition for Quality Assurance

This section presents the last of the three applications that were developed in this work: gesture interaction in an industrial environment to mark errors on painted bumpers. The challenges that were addressed with this application are two-fold. First, the application was developed to work not in a laboratory setting, but in an industrial environment. This requires a high degree of robustness and mechanisms for autonomous operation. Second, users are not interacting with displays that give direct feedback and that are stationary at well-known positions, but with real-world moving objects. These objects must consequently also be tracked and recognized in 3D.

As a difference to the applications presented in the previous sections that used video cameras, this system is based on two calibrated depth sensors, here two Kinect cameras. The 3D surface points are combined in one world coordinate system determined through calibration. This leads to a representation that is similar to surface voxels.

The application domain was quality assurance. Here, painted bumpers are inspected by workers. If any errors are detected, they must manually be documented at a separate terminal. This costs time and forces workers to shift their focus of attention from the current task to the terminal. To improve the overall process, the pointing gesture recognition system, which was introduced in Section 5.1, was applied to enable workers to directly document errors by pointing at them. This removes the need to leave the work place for documentation and, therefore, improves the overall interaction between humans and computers.

(a) 3D reconstruction

(b) Integration in quality accurance system

Figure 5.9: Pointing recognition for quality assurance. (a) shows the 3D reconstruction of the workplace with tracker bumper model (white), segmented person (green), and recognized pointing gesture (red). (b) shows the integration into the quality assurance system ANABEL where the location of the error is marked with a red dot.

In contrast to the pointing gesture recognition with known display coordinates, the bumper has to be tracked as well. The tracking is based on a set of reference points that were sampled from the bumper's CAD model. Tracking is initialized by matching 3D feature points of the bumper model with the observation. As 3D feature descriptors, Fast Point Feature Histograms [137] were used. After initialization, the bumper is tracked by using the iterative closest point algorithm for the 3D surface points.

The pointing gesture recognition follows the description in Section 5.1. The 3D points are clustered based on points that do not belong to the tracked bumper model and the main axis for recognition of the pointing gesture is specified during calibration. When the bumper remains in a resting position, workers can mark errors by pointing at them for a dwell time of one second. Example images of the gesture recognition system are shown in Figure 5.9. The system has already been installed at an industrial workshop and workers could try it in their daily work. The overall reception was very positive as it improved the general way of how humans are interacting with machines. The system was also demonstrated during the *Hannover Messe 2013* where it was one of the selected highlight exhibitions, as shown in Figure 5.10. In addition, it won the *Industriepreis 2013* in the category "research and development" of the *Initiative Mittelstand*.

(a) Exhibition at Hannover Messe 2013

(b) Installation in industrial environment ©Zentsch, Fraunhofer IOSB

Figure 5.10: Pointing gesture recognition for quality assurance in challenging environments. (a) shows the exhibition at Hannover Messe 2013 and (b) shows the trial installation in an industrial environment.

5.4 Conclusion

This concludes the section about applications that were developed in this work. A general pointing gesture recognition system was described in Section 5.1. It can be used directly on a reconstructed voxel volume as well as in combination with pose estimation. Based on this system, a general dynamic gesture recognition system was presented in Section 5.2. The example application, handwriting in mid-air, demonstrated the flexibility of this system. The high robustness of the pointing gesture recognition was shown with the example of an industrial application in Section 5.3 where workers can directly interact with bumpers in the context of quality assurance. All applications demonstrated that the real-time gesture recognition system is well suited for human-machine interaction in various contexts and application domains.

6 Conclusion

This concludes the presented thesis about human pose estimation with supervoxels. All processing steps of a system for human-computer interaction have been addressed and contributions presented, starting from sensor data acquisition to gesture interaction applications. For all steps of such a system, the thesis showed how the respective challenges can be addressed and the overall complexity reduced by segmentation techniques. The final result is a pose estimation and body tracking system with a low overall computational complexity that requires very little prior knowledge. In the following, the highlights of the various processing steps will be summarized.

The segmentation algorithms were introduced in Chapter 3. The voxel carving algorithm discussed in Section 3.1 creates a sensor-independent 3D representation of the input data. By changing the carving function, various sensor types and modalities are supported which leads to a flexibility with respect to different application areas and environments. Through the integration of occlusion maps, the approach is robust to static occlusions. In addition, it is computationally very efficient.

The superpixel segmentation algorithm in Section 3.2 provided the basis for the supervoxel segmentation algorithm in Section 3.3. Both superpixel and supervoxel algorithms achieved very good results when compared to other approaches. Further, both allow a precise control of the compactness which was demonstrated with evaluations of the presented compactness metrics. The concept of compactness was discussed in detail and the correlation to other metrics shown. Supervoxels and the presented supervoxel graphs provide the building blocks for pose estimation.

The approach introduced in Chapter 4 applies supervoxels to pictorial structures for 3D pose estimation. It achieves a significant reduction of both the search space of valid poses as well as the computational complexity and requires very little prior information. Further, through the concept of supervoxel energies, additional information can be seamlessly integrated as was shown for articulated body tracking and the integration of part detectors.

Chapter 5 introduced a gesture recognition system for both static touch and pointing gestures for interaction with large videowalls as well as dynamic gestures, as was demonstrated

with the example of handwriting recognition in mid-air. This system works both directly with voxels as well as estimated poses. The system is robust and suitable for real-world human-computer interaction applications, as was demonstrated on various exhibitions and in industrial environments.

In conclusion, this thesis investigated how both the number of input elements as well as the computational complexity can be reduced through segmentation while simultaneously improving the overall flexibility. Within this thesis, a system was developed that allows for real-time human-computer interaction. It achieves flexibility through voxel carving that can be applied to various sensor types and modalities. Based on the concept of superpixels, the size of the input data is reduced through supervoxel segmentation. Applying supervoxels as building blocks to the pictorial structures framework leads to a significant computational complexity and search space reduction that allows for real-time 3D pose estimation with little prior information. Further, the system has proven its functionality for static and dynamic gesture recognition for human-computer interaction in various laboratory and real-world applications.

Future Work

Voxel carving algorithms have been largely addressed and efficient solutions have been proposed. The major drawback of video-based voxel carving, however, are defects in the visual hull that are introduced through errors in the foreground segmentation. These errors either originate from static or dynamic occlusions or due to imperfections in the segmentation algorithms. While these issues have been addressed, image segmentation still remains a challenging topic, in particular if runtime is a limiting factor.

Superpixel segmentation has become an established image segmentation technique and there are many algorithms available that focus on different characteristics. In addition, there are various metrics available to evaluate the resulting segmentations, including the presented compactness metric. It would now be interesting to systematically apply superpixel segmentations to different application areas to evaluate which characteristics are important. In particular, the trade-off between boundary recall and compactness should be further investigated which is now possible with the presented compactness metric.

Supervoxels are relatively new compared to superpixels, in particular for sparse volume segmentations like voxel carving reconstructions. Similar to superpixels, it must now be evaluated for which applications they are most beneficial. In this context, it could also be investigated what effect compactness has on the performance of these applications.

The pose estimation approach presented in this thesis explored solutions that require little prior knowledge. In particular, no trained appearance-based part detectors were required. The system is computationally very efficient and achieves results that are comparable to other approaches. However, the estimation of the arms still poses challenges. The reason is that with a pure segmentation based approach, there are no anchors for the arm positions and due to their highly articulate nature, they can fit into various parts of the 3D reconstruction. However, possible solutions have been presented in this work, among them (simulated) part detectors. The next step could therefore be a hybrid system that uses both 3D segmentation into supervoxels to reduce the search space of possible poses as well as additional part detectors to further limit the positions of body parts. As was already shown, this information can be seamlessly integrated into the supervoxel-based pictorial structures approach to also benefit from the computational complexity reduction.

The human-computer interaction techniques presented in this work have already been applied to challenging real-world scenarios. However, the user experience can be further enhanced by improving the accuracy and robustness of the gesture recognition system. One solution could be the integration of smart interface elements that are designed to work with gesture-based input modalities that are inherently less accurate than, for example, the computer mouse. Another possible direction is the integration of hand gestures to reduce the dependence on static pointing gestures to trigger interaction events.

Publications

[1] J. Ijsselmuiden, T. Körner, A. Schick, and R. Stiefelhagen. Interaktionstechniken für große Darstellungsflächen. In *52. Fachausschusssitzung Anthropotechnik der Deutschen Gesellschaft für Luft- und Raumfahrt*, 2010.

[2] A. Schick, M. Bäuml, and R. Stiefelhagen. Improving foreground segmentations with probabilistic superpixel Markov random fields. In *Computer Vision and Pattern Recognition Workshops*, pages 27–31, 2012.

[3] A. Schick, M. Fischer, and R. Stiefelhagen. Measuring and evaluating the compactness of superpixels. In *International Conference on Pattern Recognition (Best Scientific Paper Award)*, pages 930–934, 2012.

[4] A. Schick, M. Fischer, and R. Stiefelhagen. An evaluation of the compactness of superpixels. *Pattern Recognition Letters*, 43:71–80, 2014.

[5] A. Schick, D. Morlock, C. Amma, T. Schultz, and R. Stiefelhagen. Vision-Based Handwriting Recognition for Unrestricted Text Input in Mid-Air. In *International Conference on Multimodal Interaction*, 2012.

[6] A. Schick and R. Stiefelhagen. Real-Time GPU-Based Voxel Carving with Systematic Occlusion Handling. In *DAGM Symposium on Pattern Recognition*, pages 372–381, 2009.

[7] A. Schick and R. Stiefelhagen. Evaluating image segments by applying the description length to sets of superpixels. In *International Conference on Computer Vision Workshops*, pages 1394–1401, 2011.

[8] A. Schick, F. van De Camp, J. Ijsselmuiden, and R. Stiefelhagen. Extending Touch: Towards Interaction with Large-Scale Surfaces. In *International Conference on Interactive Tabletops and Surfaces*, pages 117–124, 2009.

[9] F. van de Camp, A. Schick, and R. Stiefelhagen. How to Click in Mid-Air. In *International Conference on Human-Computer Interaction*, 2013.

[10] M. Voit, F. van de Camp, J. Ijsselmuiden, A. Schick, and R. Stiefelhagen. Visuelle Perzeption für die multimodale Mensch-Maschine-Interaktion in und mit aufmerksamen Räumen. *at - Automatisierungstechnik*, 61(11), 2013.

[11] D. Weikersdorfer, A. Schick, and D. Cremers. Depth-Adaptive Supervoxels for RGB-D Video Segmentation. In *International Conference on Image Processing*, 2013.

Supervised Student Theses

[12] S. Bittel. Virtuelle Interaktionshülle für druckbasierte Gestenerkennung. *Bachelor Thesis*, Karlsruhe Institut für Technologie. Institut für Anthropomatik. 2012.

[13] V. Henne. Hand Gesture Recognition with a Depth-sensing Camera. *Bachelor Thesis*, Karlsruhe Institute of Technology. Institute for Anthropomatics. 2011.

[14] C. Johner. Voxel Coloring und Color Carving. *Studienarbeit*, Karlsruhe Institut für Technologie. Institut für Anthropomatik. 2010.

[15] D. Morlock. Gestenbasierte Texteingabe an großen Displays. *Diplomarbeit*, Karlsruhe Institut für Technologie. Institut für Anthropomatik. 2011.

[16] O. Rösch. Empirische Evaluation von Kalibrierobjektposen auf die Kamerakalibrierung. *Studienarbeit*, Karlsruhe Institut für Technologie. Institut für Anthropomatik. 2012.

[17] O. Rösch. Framework zur automatischen Registrierung eines Multi-Kinect-Systems. *Diplomarbeit*, Karlsruhe Institut für Technologie. Institut für Anthropomatik. 2013.

[18] M. Tirpitz. Berechnung der visuellen Hülle mit mehreren Tiefensensoren. *Studienarbeit*, Karlsruhe Institut für Technologie. Institut für Anthropomatik. 2012.

Bibliography

[19] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC Superpixels. In *EPFL Technical Report no. 149300*, 2010.

[20] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk. SLIC superpixels compared to state-of-the-art superpixel methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–82, 2012.

[21] A. Agarwal and B. Triggs. 3D human pose from silhouettes by relevance vector regression. In *Computer Vision and Pattern Recognition*, volume 2, pages 882–888, 2004.

[22] A. Agarwal and B. Triggs. Recovering 3D human pose from monocular images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):44–58, 2006.

[23] J. Aggarwal and Q. Cai. Human Motion Analysis: A Review. *Computer Vision and Image Understanding*, 73(3):428–440, 1999.

[24] J. Aggarwal, Q. Cai, W. Liao, and B. Sabata. Articulated and elastic non-rigid motion: a review. In *IEEE Workshop on Motion of Non-Rigid and Articulated Objects*, pages 2–14, 1994.

[25] J. Aloimonos. Shape from texture. *Biological Cybernetics*, 58(5):345–360, 1988.

[26] S. Amin, M. Andriluka, M. Rohrbach, and B. Schiele. Multi-view Pictorial Structures for 3D Human Pose Estimation. In *British Machine Vision Conference*, 2013.

[27] C. Amma, D. Gehrig, and T. Schultz. Airwriting Recognition Using Wearable Motion Sensors. In *Augmented Human International Conference*, pages 10:1–10:8, 2010.

[28] B. Andres, U. Köthe, M. Helmstaedter, W. Denk, and F. A. Hamprecht. Segmentation of SBFSEM Volume Data of Neural Tissue by Hierarchical Classification. In *DAGM Symposium on Pattern Recognition*, pages 142–152, 2008.

[29] M. Andriluka, S. Roth, and B. Schiele. Pictorial Structures Revisited: People Detection and Articulated Pose Estimation. In *Computer Vision and Pattern Recognition*, pages 1014–1021, 2009.

[30] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[31] A. Ayvaci and S. Soatto. Motion segmentation with occlusions on the superpixel graph. In *International Conference on Computer Vision Workshops*, pages 727–734, 2009.

[32] P. Azad, T. Asfour, and R. Dillmann. Robust real-time stereo-based markerless human motion capture. In *International Conference on Humanoid Robots*, pages 700–707, 2008.

[33] O. Barnich and M. Van Droogenbroeck. ViBe: a universal background subtraction algorithm for video sequences. *IEEE Transactions on Image Processing*, 20(6):1709–24, 2011.

[34] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. Review and evaluation of commonly-implemented background subtraction algorithms. In *International Conference on Pattern Recognition*, pages 1–4, 2008.

[35] Y. Benezeth, P.-M. Jodoin, B. Emile, H. Laurent, and C. Rosenberger. Comparative study of background subtraction algorithms. *Journal of Electronic Imaging*, 19(3):1—-12, 2010.

[36] K. Berger, K. Ruhl, Y. Schroeder, C. Bruemmer, A. Scholz, and M. Magnor. Markerless Motion Capture using multiple Color-Depth Sensors. In *Vision, Modeling, and Visualization*, pages 317–324, 2011.

[37] O. Bernier, P. Cheung-Mon-Chan, and A. Bouguet. Fast nonparametric belief propagation for real-time stereo articulated body tracking. *Computer Vision and Image Understanding*, 113(1):29–47, 2009.

[38] L. Bourdev, S. Maji, T. Brox, and J. Malik. Detecting People Using Mutually Consistent Poselet Activations. In *European Conference on Computer Vision*, pages 168–181, 2010.

[39] L. Bourdev and J. Malik. Poselets: Body part detectors trained using 3D human pose annotations. In *International Conference on Computer Vision*, pages 1365–1372, 2009.

[40] M. Brand. Shadow puppetry. In *International Conference on Computer Vision*, volume 2, pages 1237–1244, 1999.

[41] J. Brauer, W. Gong, J. Gonzàlez, and M. Arens. On the effect of temporal information on monocular 3d human pose estimation. In *International Conference on Computer Vision Workshops*, pages 906–913, 2011.

[42] J. Brauer, W. Hübner, and M. Arens. Generative 2D and 3D Human Pose Estimation with Vote Distributions. In *International Symposium on Visual Computing*, pages 470–481, 2012.

[43] E. Bribiesca. A measure of compactness for 3D shapes. *Computers & Mathematics with Applications*, 40(10):1275–1284, 2000.

[44] E. Bribiesca. An easy measure of compactness for 2D and 3D shapes. *Pattern Recognition*, 41(2):543–554, 2008.

[45] M. Burenius, J. Sullivan, and S. Carlsson. 3D Pictorial Structures for Multiple View Articulated Pose Estimation. In *Conference on Computer Vision and Pattern Recognition*, pages 3618–3625, 2013.

[46] F. Caillette and T. Howard. Real-Time Markerless Human Body Tracking Using Colored Voxels and 3-D Blobs. In *ACM International Symposium on Mixed and Augmented Reality*, pages 266–267, 2004.

[47] F. Caillette and T. Howard. Real-Time Markerless Human Body Tracking with 3-D Voxel Reconstruction. In *British Machine Vision Conference 2004*, 2004.

[48] C. Canton-Ferrer, J. Casas, and M. Pardas. Voxel based annealed particle filtering for markerless 3D articulated motion capture. In *3DTV Conference: The True Vision - Capture, Transmission and Display of 3D Video*, pages 1–4, 2009.

[49] S. K. Card, A. Newell, and T. P. Moran. *The Psychology of Human-Computer Interaction*. L. Erlbaum Associates Inc., 1983.

[50] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel. Free-Viewpoint Video of Human Actors. In *International Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH*, pages 569—-577, 2003.

[51] S. Y. Cheng and M. M. Trivedi. Articulated Human Body Pose Inference from Voxel Data Using a Kinematically Constrained Gaussian Mixture Model. In *Computer Vision and Pattern Recognition Workshops*, 2007.

[52] G. K. Cheung, S. Baker, and T. Kanade. Shape-from-silhouette of articulated objects and its use for human body kinematics estimation and motion capture. In *Computer Vision and Pattern Recognition*, pages 77–84, 2003.

[53] G. K. Cheung, T. Kanade, J.-Y. Bouguet, and M. Holler. A real time system for robust 3D voxel reconstruction of human motions. In *Computer Vision and Pattern Recognition*, pages 714–720, 2000.

[54] R. Cipolla, T. Drummond, and D. Robertson. Camera Calibration from Vanishing Points in Image of Architectural Scenes. In *British Machine Vision Conference*, pages 382–391, 1999.

[55] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[56] W. B. Culbertson, T. Malzbender, and G. G. Slabaugh. Generalized Voxel Coloring. In *International Workshop on Vision Algorithms: Theory and Practice (in conjunction with ICCV)*, volume 1883 of *Lecture Notes in Computer Science*, pages 100–115, 1999.

[57] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

[58] Q. Delamarre and O. Faugeras. 3D Articulated Models and Multiview Tracking with Physical Forces. *Computer Vision and Image Understanding*, 81(3):328–357, 2001.

[59] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *Computer Vision and Pattern Recognition*, volume 2, pages 126–133, 2000.

[60] R. Drillis and R. Contini. Body segment parameters. Technical Report No. 1166.03. New York University, School of Engineering and Science. Technical report, 1966.

[61] A. Elgammal, D. Harwood, and L. Davis. Non-parametric Model for Background Subtraction. In *European Conference on Computer Vision*, pages 751–767, 2000.

[62] O. D. Faugeras, Q.-T. Luong, and S. J. Maybank. Camera Self-Calibration: Theory and Experiments. In *European Conference on Computer Vision*, pages 321–334, 1992.

[63] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient Graph-Based Image Segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[64] P. F. Felzenszwalb and D. P. Huttenlocher. Pictorial Structures for Object Recognition. *International Journal of Computer Vision*, 61(1):55–79, 2005.

[65] V. Ferrari, M. Marín-Jimenéz, and A. Zisserman. Progressive search space reduction for human pose estimation. In *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[66] M. A. Fischler and R. A. Elschlager. The Representation and Matching of Pictorial Structures. *IEEE Transactions on Computers*, C-22(1):67–92, 1973.

[67] J.-S. Franco and E. Boyer. Exact polyhedral visual hulls. In *British Machine Vision Conference*, pages 329–338, 2003.

[68] R. C. Fromuth and M. B. Parkinson. Predicting 5th and 95th percentile anthropometric segment lengths from population stature. In *ASME International Design Engineering Technical Conferences. DETC2008-50091.*, 2008.

[69] B. Fulkerson, A. Vedaldi, and S. Soatto. Class segmentation and object localization with superpixel neighborhoods. In *International Conference on Computer Vision*, pages 670–677, 2009.

[70] V. Ganapathi, C. Plagemann, D. Koller, and S. Thrun. Real time motion capture using a single time-of-flight camera. In *Computer Vision and Pattern Recognition*, pages 755–762, 2010.

[71] D. M. Gavrila. The Visual Analysis of Human Movement: A Survey. *Computer Vision and Image Understanding*, 73(1):82–98, 1999.

[72] N. Gordon, D. Salmond, and A. Smith. Novel approach to nonlinear/non-Gaussian Bayesian state estimation. *Radar and Signal Processing, IEE Proceedings F*, 140(2):107–113, 1993.

[73] L. Guan, J.-S. Franco, and M. Pollefeys. 3D Occlusion Inference from Silhouette Cues. In *Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[74] L. Guan, S. Sinha, J.-S. Franco, and M. Pollefeys. Visual Hull Construction in the Presence of Partial Occlusion. In *International Symposium on 3D Data Processing, Visualization, and Transmission*, pages 413–420, 2006.

[75] N. Hasler, B. Rosenhahn, T. Thormählen, M. Wand, J. Gall, and H.-P. Seidel. Markerless Motion Capture with unsynchronized moving cameras. In *Computer Vision and Pattern Recognition*, pages 224–231, 2009.

[76] J. Heikkila. Geometric camera calibration using circular control points. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(10):1066–1077, 2000.

[77] M. Hofmann and D. M. Gavrila. Multi-view 3D Human Pose Estimation in Complex Environment. *International Journal of Computer Vision*, 96(1):103–124, 2012.

[78] M. B. Holte, C. Tran, M. M. Trivedi, and T. B. Moeslund. Human Pose Estimation and Activity Recognition From Multi-View Videos: Comparative Explorations of Recent Developments. *IEEE Journal of Selected Topics in Signal Processing*, 6(5):538–552, 2012.

[79] B. K. Horn. Shape from Shading: A Method for Obtaining the Shape of a Smooth Opaque Object from One View. In *Technical Report 232. MIT Artificial Intelligence Laboratory*, 1970.

[80] N. R. Howe. Silhouette Lookup for Automatic Pose Tracking. In *Computer Vision and Pattern Recognition Workshop*, pages 15–22, 2004.

[81] Y. Huang and T. S. Huang. Model-based human body tracking. In *International Conference on Pattern Recognition*, volume 1, pages 552–555, 2002.

[82] M. Isard and A. Blake. CONDENSATION - Conditional Density Propagation forVisual Tracking. *International Journal of Computer Vision*, 29(1):5–28, 1998.

[83] S. Iwasawa, K. Ebihara, J. Ohya, and S. Morishima. Real-time estimation of human body posture from monocular thermal images. In *Computer Vision and Pattern Recognition*, pages 15–20, 1997.

[84] X. Ji and H. Liu. Advances in View-invariant Human Motion Analysis: A Review. *Transactions on Systems, Man, and Cybernetics - Part C*, 40(1):13–24, 2010.

[85] I. A. Kakadiaris and D. Metaxas. Model-based estimation of 3D human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1453–1459, 2000.

[86] A. Kanaujia, N. Kittens, and N. Ramanathan. Part Segmentation of Visual Hull for 3D Human Pose Estimation. In *Computer Vision and Pattern Recognition Workshops*, pages 542–549, 2013.

[87] R. Kehl, M. Bray, and L. Van Gool. Full Body Tracking from Multiple Views Using Stochastic Sampling. In *Computer Vision and Pattern Recognition*, pages 129–136, 2005.

[88] R. Kehl and L. V. Gool. Markerless tracking of complex human motions from multiple views. *Computer Vision and Image Understanding*, 104(2):190–209, 2006.

[89] H. Kim, R. Sakamoto, I. Kitahara, N. Orman, T. Toriyama, and K. Kogure. Compensated Visual Hull for Defective Segmentation and Occlusion. In *International Conference on Artificial Reality and Telexistence*, pages 210–217, 2007.

[90] P. Kohli, L. Ladický, and P. H. Torr. Robust Higher Order Potentials for Enforcing Label Consistency. *International Journal of Computer Vision*, 82(3):302–324, 2009.

[91] A. Ladikos, S. Benhimane, and N. Navab. Efficient visual hull computation for real-time 3D reconstruction using CUDA. In *Computer Vision and Pattern Recognition Workshops*, pages 1–8, 2008.

[92] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, 1994.

[93] B. Leibe, A. Leonardis, and B. Schiele. Combined object categorization and segmentation with an implicit shape model. *Workshop on Statistical Learning in Computer Vision, ECCV*, 2004.

[94] B. Leibe, A. Leonardis, and B. Schiele. Robust Object Detection with Interleaved Categorization and Segmentation. *International Journal of Computer Vision*, 77(1-3):259–289, 2008.

[95] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi. TurboPixels: Fast Superpixels Using Geometric Flows. *Transactions on Pattern Analysis and Machine Intelligence*, 31(12):2290–2297, 2009.

[96] M. Li, M. Magnor, and H.-P. Seidel. Hardware-Accelerated Visual Hull Reconstruction and Rendering. In *Proceedings of Graphics Interface*, pages 65—-71, 2003.

[97] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa. Entropy rate superpixel segmentation. In *Computer Vision and Pattern Recognition*, pages 2097–2104, 2011.

[98] A. Lucchi, K. Smith, R. Achanta, G. Knott, and P. Fua. Supervoxel-based segmentation of mitochondria in em image stacks with learned shape features. *IEEE Transactions on Medical Imaging*, 31(2):474–86, 2012.

[99] A. Lucchi, K. Smith, R. Achanta, V. Lepetit, and P. Fua. A Fully Automated Approach to Segmentation of Irregularly Shaped Cellular Structures in EM Images. In *International Conference on Medical Image Computing and Computer Assisted Intervention*, 2010.

[100] J. Luck, D. Small, and C. Q. Little. Real-Time Tracking of Articulated Human Models Using a 3D Shape-from-Silhouette Method. In *International Workshop on Robot Vision*, pages 19–26, 2001.

[101] J. Malik, S. Belongie, T. Leung, and J. Shi. Contour and Texture Analysis for Image Segmentation. *International Journal of Computer Vision*, 43(1):7–27, 2001.

[102] D. Martin, C. Fowlkes, D. Tal, and J. Malik. A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In *International Conference on Computer Vision*, pages 416–423, 2001.

[103] W. N. Martin and J. K. Aggarwal. Volumetric Descriptions of Objects from Multiple Views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-5(2):150–158, 1983.

[104] W. Matusik, C. Buehler, and L. McMillan. Polyhedral visual hulls for real-time rendering. In *Eurographics Conference on Rendering*, pages 115–126, 2001.

[105] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-Based Visual Hulls. In *Annual Conference on Computer Graphics and Interactive Techniques*, pages 369–374, 2000.

[106] D. J. R. Meagher. *Octree Encoding: a New Technique for the Representation, Manipulation and Display of Arbitrary 3-D Objects by Computer*. 1980.

[107] I. Mikić, M. Trivedi, E. Hunter, and P. Cosman. Articulated body posture estimation from multi-camera voxel data. In *Computer Vision and Pattern Recognition*, volume 1, pages 455–460, 2001.

[108] I. Mikić, M. Trivedi, E. Hunter, and P. Cosman. Human Body Model Acquisition and Tracking Using Voxel Data. *International Journal of Computer Vision*, 53(3):199–223, 2003.

[109] T. B. Moeslund and E. Granum. A Survey of Computer Vision-Based Human Motion Capture. *Computer Vision and Image Understanding*, 81(3):231—-268, 2001.

[110] T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104(2-3):90–126, 2006.

[111] A. P. Moore, S. J. Prince, and J. Warrell. "Lattice Cut" - Constructing superpixels using layer constraints. In *Computer Vision and Pattern Recognition*, pages 2117–2124, 2010.

[112] A. P. Moore, S. J. Prince, J. Warrell, U. Mohammed, and G. Jones. Superpixel lattices. In *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[113] G. Mori. Guiding model search using segmentation. In *International Conference on Computer Vision*, pages 1417–1423 Vol. 2, 2005.

[114] G. Mori, A. A. Efros, and J. Malik. Recovering Human Body Configurations: Combining Segmentation and Recognition. In *Computer Vision and Pattern Recognition*, pages 326–333, 2004.

[115] J. Müller and M. Arens. Human Pose Estimation with Implicit Shape Models. In *International Workshop on Analysis and Retrieval of Tracked Events and Motion in Imagery Streams*, pages 9–14, 2010.

[116] R. M. Murray, Z. Li, and S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc., 1st edition, 1994.

[117] S. K. Nayar and Y. Nakagawa. Shape from focus. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):824–831, 1994.

[118] K. Nickel, E. Seemann, and R. Stiefelhagen. 3D-tracking of head and hands for pointing gesture recognition in a human-robot interaction scenario. In *International Conference on Automatic Face and Gesture Recognition*, pages 565–570, 2004.

[119] K. Nickel and R. Stiefelhagen. Pointing Gesture Recognition Based on 3D-tracking of Face, Hands and Head Orientation. In *International Conference on Multimodal Interfaces*, pages 140–146, 2003.

[120] S. Nowozin, P. V. Gehler, and C. H. Lampert. On parameter learning in CRF-based approaches to object class image segmentation. In *European Conference on Computer vision*, pages 98–111, 2010.

[121] J. Papon, A. Abramov, M. Schoeler, and F. Wörgötter. Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds. In *Computer Vision and Pattern Recognition*, pages 2027–2034, 2013.

[122] F. J. Perales and J. Torres. A system for human motion matching between synthetic and real images based on a biomechanic graphical model. In *Workshop on Motion of Non-rigid and Articulated Objects*, pages 83–88, 1994.

[123] F. Perbet and A. Maki. Homogeneous superpixels from random walks. In *Conference on Machine Vision Applications*, pages 26—-30, 2011.

[124] F. Perbet, B. Stenger, and A. Maki. Homogeneous Superpixels from Markov RandomWalks. *IEICE Transactions on Information and Systems*, E95-D(7):1740–1748, 2012.

[125] M. Piccardi. Background subtraction techniques: a review. In *International Conference on Systems, Man and Cybernetics*, volume 4, pages 3099–3104, 2004.

[126] R. Plamondon and S. N. Srihari. Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[127] R. Plänkers and P. Fua. Tracking and Modeling People in Video Sequences. *Computer Vision and Image Understanding*, 81(3):285–302, 2001.

[128] G. Polya. *Mathematics and Plausible Reasoning, Volume 1: Induction and Analogy in Mathematics*. 1990.

[129] R. Poppe. Vision-based human motion analysis: An overview. *Computer Vision and Image Understanding*, 108(1):4–18, 2007.

[130] R. Poppe. A Survey on Vision-based Human Action Recognition. *Image and Vision Computing*, 28(6):976–990, 2010.

[131] M. Potmesil. Generating octree models of 3D objects from their silhouettes in a sequence of images. *Computer Vision, Graphics, and Image Processing*, 40(1):1–29, 1987.

[132] T. Rabbani, F. van den Heuvel, and G. Vosselmann. Segmentation of point clouds using smoothness constraint. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 36(5), 2006.

[133] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[134] V. Ramakrishna, T. Kanade, and Y. Sheikh. Tracking Human Pose by Tracking Symmetric Parts. In *Computer Vision and Pattern Recognition*, pages 3728–3735, 2013.

[135] R. Rosales, M. Siddiqui, J. Alon, and S. Sclaroff. Estimating 3D body pose using uncalibrated cameras. In *Computer Vision and Pattern Recognition.*, pages 821–827, 2001.

[136] C. Rother, V. Kolmogorov, and A. Blake. Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Transactions on Graphics*, 23(3):309—-314, 2004.

[137] R. B. Rusu, N. Blodow, and M. Beetz. Fast Point Feature Histograms (FPFH) for 3D registration. In *International Conference on Robotics and Automation*, pages 3212–3217, 2009.

[138] R. B. Rusu and S. Cousins. 3D is here: Point Cloud Library (PCL). In *International Conference on Robotics and Automation*, pages 1–4, 2011.

[139] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz. Towards 3D Point cloud based object maps for household environments. *Robotics and Autonomous Systems*, 56(11):927–941, 2008.

[140] Y. Sagawa, M. Shimosaka, T. Mori, and T. Sato. Fast online human pose estimation via 3D voxel data. In *International Conference on Intelligent Robots and Systems*, pages 1034–1040, 2007.

[141] L. A. Schwarz, A. Mkhitaryan, D. Mateus, and N. Navab. Estimating human 3D pose from Time-of-Flight images based on geodesic distances and optical flow. In *Automatic Face & Gesture Recognition and Workshops*, pages 700–706, 2011.

[142] S. M. Seitz and C. R. Dyer. Photorealistic Scene Reconstruction by Voxel Coloring. In *Conference on Computer Vision and Pattern Recognition*, pages 1067–1073, 1997.

[143] S. M. Seitz and C. R. Dyer. Photorealistic Scene Reconstruction by Voxel Coloring. *International Journal of Computer Vision*, 35(2):151–173, 1999.

[144] K. Shanmukh and A. K. Pujari. Volume intersection with optimal set of directions. *Pattern Recognition Letters*, 12(3):165–170, 1991.

[145] J. Shi and J. Malik. Normalized Cuts and Image Segmentation. *Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2000.

[146] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, and A. Blake. Real-time human pose recognition in parts from single depth images. In *Computer Vision and Pattern Recognition*, pages 1297–1304, 2011.

[147] L. Sigal, A. O. Balan, and M. J. Black. HumanEva : Synchronized Video and Motion Capture Dataset and Baseline Algorithm for Evaluation of Articulated Human Motion. *International Journal of Computer Vision*, 87(1-2):4–27, 2010.

[148] L. Sigal and M. J. Black. HumanEva: Synchronized Video and Motion Capture Dataset for Evaluation of Articulated Human Motion. In *Technical Report CS-06-08, Brown University*, number September, 2006.

[149] L. Sigal, B. H. Sigelman, M. Isard, and M. J. Black. Attractive People : Assembling Loose-Limbed Models using Non-parametric Belief Propagation. *Advances in Neural Information Processing Systems*, 16:1539–1546, 2003.

[150] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor Segmentation and Support Inference from RGBD Images. In *European Conference on Computer Vision*, pages 746–760, 2012.

[151] C. Sminchisescu and B. Triggs. Estimating Articulated Human Motion with Covariance Scaled Sampling. *International Journal of Robotics Research*, 22(6):371–393, 2003.

[152] J. Smisek, M. Jancosek, and T. Pajdla. 3D with Kinect. In *International Conference on Computer Vision Workshops*, pages 1154–1160, 2011.

[153] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition*, pages 246–252, 1999.

[154] P. F. Sturm and S. J. Maybank. On plane-based camera calibration: A general algorithm, singularities, applications. In *Computer Vision and Pattern Recognition*, pages 432–437, 1999.

[155] R. Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Underst.*, 58(1):23–32, 1993.

[156] R. Szeliski and S. B. Kang. Recovering 3D shape and motion from image streams using nonlinear least squares. In *Computer Vision and Pattern Recognition*, pages 752–753, 1993.

[157] C. J. Taylor. Reconstruction of Articulated Objects from Point Correspondences in a Single Uncalibrated Image. *Computer Vision and Image Understanding*, 80(3):349–363, 2000.

[158] J. Taylor, J. Shotton, T. Sharp, and A. Fitzgibbon. The Vitruvian manifold: Inferring dense correspondences for one-shot human pose estimation. In *Computer Vision and Pattern Recognition*, pages 103–110, 2012.

[159] C. Tomasi and T. Kanade. Shape and motion from image streams under orthography: a factorization method. *International Journal of Computer Vision*, 9(2):137–154, 1992.

[160] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: principles and practice of background maintenance. In *International Conference on Computer Vision*, pages 255–261, 1999.

[161] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses. *IEEE Journal on Robotics and Automation*, 3(4):323–344, 1987.

[162] N. van der Aa, X. Luo, G. Giezeman, R. Tan, and R. Veltkamp. UMPM benchmark: A multi-person dataset with synchronized video and motion capture data for evaluation of articulated human motion and interaction. In *International Conference on Computer Vision Workshops*, pages 1264–1269, 2011.

[163] O. Veksler, Y. Boykov, and P. Mehrani. Superpixels and Supervoxels in an Energy Optimization Framework. In *European Conference on Computer Vision*, pages 211–224, 2010.

[164] L. Vincent and P. Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[165] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition*, pages 511–518, 2001.

[166] D. Vlasic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics*, 27(3), 2008.

[167] C. von Hardenberg and F. Bérard. Bare-hand Human-computer Interaction. In *Workshop on Perceptive User Interfaces*, PUI '01, pages 1–8, 2001.

[168] S. Wachter and H.-H. Nagel. Tracking Persons in Monocular Image Sequences. *Computer Vision and Image Understanding*, 74(3):174–192, 1999.

[169] A. Waibel and R. Stiefelhagen, editors. *Computers in the Human Interaction Loop*. Springer Verlag, London, 2009.

[170] L. Wang, W. Hu, and T. Tan. Recent developments in human motion analysis. *Pattern Recognition*, 36(3):585–601, 2003.

[171] S. Wang, H. Lu, F. Yang, and M.-H. Yang. Superpixel tracking. In *International Conference on Computer Vision*, pages 1323–1330, 2011.

[172] D. Weikersdorfer, D. Gossow, and M. Beetz. Depth-adaptive superpixels. In *International Conference on Pattern Recognition*, pages 2087–2090, 2012.

[173] D. Weinland, R. Ronfard, and E. Boyer. Free Viewpoint Action Recognition Using Motion History Volumes. *Computer Vision and Image Understanding*, 104(2):249–257, 2006.

[174] D. Weinland, R. Ronfard, and E. Boyer. A Survey of Vision-based Methods for Action Representation, Segmentation and Recognition. *Computer Vision and Image Understanding*, 115(2):224–241, 2011.

[175] J. Weng, P. Cohen, and M. Herniou. Camera calibration with distortion models and accuracy evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(10):965–980, 1992.

[176] N. Werghi. Segmentation and Modeling of Full Human Body Shape From 3-D Scan Data: A Survey. *Transactions on Systems, Man, and Cybernetics - Part C*, 37(6):1122–1136, 2007.

[177] A. D. Wilson and H. Benko. Combining multiple depth cameras and projectors for interactions on, above and between surfaces. In *ACM Symposium on User Interface Software and Technology*, pages 273–282, 2010.

[178] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: real-time tracking of the human body. In *International Conference on Automatic Face and Gesture Recognition*, pages 51–56, 1996.

[179] C. R. Wren, A. Azarbayejani, T. Darrell, and A. P. Pentland. Pfinder: real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.

[180] S. Xiang, C. Pan, F. Nie, and C. Zhang. TurboPixel Segmentation Using Eigen-Images. *IEEE Transactions on Image Processing*, 19(11):3024–3034, 2010.

[181] Xiaofeng Ren and J. Malik. Learning a classification model for segmentation. In *International Conference on Computer Vision*, pages 10–17, 2003.

[182] C. Xu and J. J. Corso. Evaluation of super-voxel methods for early video processing. In *Computer Vision and Pattern Recognition*, pages 1202–1209, 2012.

[183] G. Zeng, P. Wang, J. Wang, R. Gan, and H. Zha. Structure-sensitive superpixels via geodesic distance. In *International Conference on Computer Vision*, pages 447–454, 2011.

[184] R. Zhang, P.-S. Tsai, J. E. Cryer, and M. Shah. Shape-from-shading: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.

[185] Y. Zhang, R. Hartley, J. Mashford, and S. Burn. Superpixels via pseudo-Boolean optimization. In *International Conference on Computer Vision*, pages 1387–1394, 2011.

[186] Z. Zhang. A Flexible New Technique for Camera Calibration. In *Technical Report MSR-TR-98-71*, number 11, 1998.

[187] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[188] J. Ziegler, K. Nickel, and R. Stiefelhagen. Tracking of the Articulated Upper Body on Multi-View Stereo Image Sequences. In *Computer Vision and Pattern Recognition*, pages 774–781, 2006.

[189] Z. Zivkovic. Improved adaptive Gaussian mixture model for background subtraction. In *International Conference on Pattern Recognition*, pages 28–31 Vol.2, 2004.