FACULTY OF ENGINEERING
AND ARCHITECTURE

3D Reconstruction of Persons and Objects Using Multiple Cameras with Overlapping Views in the Presence of Occlusion

Maarten Slembrouck

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Engineering

Supervisors
Prof. Peter Veelaert, PhD - Prof. Wilfried Philips, PhD

Department of Telecommunications and Information Processing
Faculty of Engineering and Architecture, Ghent University

October 2021

GHENT
UNIVERSITY

FACULTY OF ENGINEERING
AND ARCHITECTURE

3D Reconstruction of Persons and Objects Using Multiple Cameras with Overlapping Views in the Presence of Occlusion

Maarten Slembrouck

Doctoral dissertation submitted to obtain the academic degree of
Doctor of Engineering

Supervisors
Prof. Peter Veelaert, PhD - Prof. Wilfried Philips, PhD

Department of Telecommunications and Information Processing
Faculty of Engineering and Architecture, Ghent University

October 2021

GHENT
UNIVERSITY

ISBN 978-94-6355-530-2
NUR 958
Wettelijk depot: D/2021/10.500/78

Members of the Examination Board

Chair

Prof. Patrick De Baets, PhD, Ghent University

Other members entitled to vote

Prof. Em. Dirk De Clercq, PhD, Ghent University
Prof. Peter Lambert, PhD, Ghent University
Prof. Hiep Luong, PhD, Ghent University
Jürgen Slowack, PhD, Barco
Prof. Steven Verstockt, PhD, Ghent University

Supervisors

Prof. Peter Veelaert, PhD, Ghent University
Prof. Wilfried Philips, PhD, Ghent University

Acknowledgements

Needless to say, this work would not have been accomplished without the help and support of family, friends and colleagues. Therefore, I want to take a moment to express special thanks to several people.

I would like to thank my supervisors, prof. Peter Veelaert and prof. Wilfried Philips for giving me the opportunity to pursue my doctoral degree in the Image Processing and Interpretation group. In particular, I want to say thank you for reviewing the texts I have written. The criticism, hard at times, certainly improved the quality of the work. I also want to thank the jury members em. prof. Dirk De Clercq, prof. Peter Lambert, prof. Hiep Luong, prof. Steven Verstockt and dr. ir. Jürgen Slowack to take the time to read my dissertation and to provide valuable feedback.

A special thank you goes to my parents, Mieke Vanhollemeersch and Johan Slembrouck, for giving me the opportunity to study engineering in Ghent and to support me unconditionally in all my endeavours.

I also want to thank my (ex-)colleagues at TELIN. Without them, the past years would not have been the same. Numerous lunch break discussions about silly subjects brightened up the work-life. Unfortunately, social contact was significantly reduced during the Corona pandemic, but I am confident that we will pick this up again.

I spent much time outside of work on ice rinks to be involved in my true passion, Short Track. It is a welcome distraction from work at Ghent University during the many training sessions and competitions. Therefore, I thank all my friends that are involved in this beautiful sport.

And last but not least, I would like to thank my girlfriend, Fabienne Coremans, to keep up with me during times when I kept rambling on about several research problems, even though she has no reason to care about them.

Ghent, October 7, 2021
Maarten Slembrouck

Table of Contents

Samenvatting

Wanneer een camera zijn omgeving vastlegt, gaat de 3D-structuur verloren tijdens het projectieproces op de beeldsensor. Bij computervisie wordt onder andere onderzocht hoe computers de gecapteerde ongeving op hoog niveau kunnen begrijpen op. Vaak is het herwinnen van een begrip van deze 3D-structuren een essentiële stap om de ongeving correct te interpreteren. Het observeren van de omgeving vanuit meerdere gezichtspunten helpt om deze beter te begrijpen dan wanneer slechts één gezichtspunt wordt gebruikt. Door de informatie in de beelden van verschillende gezichtspunten te fuseren, is het mogelijk om een 3D-reconstructie te maken van de objecten in de omgeving.

Vanuit meerdere camerabeelden kunnen 3D-modellen van objecten gemaakt worden en specifieke eigenschappen gemeten worden, zoals hun volumeverdeling. Een typische use case is bijvoorbeeld het monitoren van het groeiproces van een plant door op regelmatige tijdstippen deze plant te reconstrueren om zo de invloed van zonlicht en meststoffen te onderzoeken. Omdat computers steeds krachtiger worden, kunnen we zelfs 3D-objecten reconstrueren in realtime. De volumetrische informatie van objecten in de ongeving kan worden gebruikt voor het volgen van de bewegingen van mensen in de ongeving of zelfs voor het analyseren van verkeer.

Augmented en virtual reality (AR/VR) vormen een ander toepassingsgebied voor de technieken die voorgesteld worden in dit boek. Omdat dit soort toepassingen sterk afhankelijk zijn van de bestaande 3D-omgeving. Huidige oplossingen tracken niet volledige beweging van het lichaam van een gebruiker, maar enkel van specifieke apparaten die worden gedragen door de gebruiker, bv. headsets en handcontrollers. Het tracken van het hele lichaam kan helpen om een meeslependere gebruikerservaring te creëren. Zo'n applicatie vereist zowel volumetrische reconstructie en schatting van de lichaamshouding.

In realistische omgevingen kunnen de camera's vaak niet het volledig

object of de volledige persoon vastleggen vanwege obstakels tussen de locatie van het object of de persoon en de camera. Dit fenomeen wordt occlusie genoemd. Occlusies zijn afhankelijk van het perspectief van de camera. Wanneer een object in een bepaald aanzicht geoccludeerd is, betekent dit niet noodzakelijkerwijs dit ook het geval is vanuit een ander gezichtspunt. Occlusie gebeurt voornamelijk op drie manieren. Objecten bewegen achter statische obstakels in de scène (statische occlusie), bv. een persoon die achter een boom loopt, of door bewegende objecten die een ander object tijdelijk verbergen (dynamische occlusie), bv. een auto die voor een voetganger passeert, of doordat delen van het object ander delen van hetzelfde object vebergen (zelfocclusie), bv. iemands arm voor zijn romp.

State-of-the-art methoden gaan niet goed om met occlusie. De 3D-reconstructie van een object ontbreekt in zo'n geval typisch delen die niet zichtbaar zijn vanuit alle gezichtspunten. De meeste van deze 3D-reconstructiemethoden gaan er immers van uit dat alle camera's het volledig object kunnen observeren. Bijvoorbeeld, wanneer een persoon naast een tafel staat, kan de reconstructie een been missen omdat de tafel het been in ten minste één aanzicht verbergt. Om de impact van occlusie op de 3D-reconstructie te verminderen, is het essentieel om voldoende gezichtspunten te hebben waarin dezelfde delen van het object niet telkens worden geoccludeerd.

Om de uitdagingen bij het reconstrueren van 3D-geometrie vanuit beelden het hoofd te bieden, nemen toepassingen die een hoge nauwkeurigheid vereisen vaak hun toevlucht tot gespecialiseerde hardware of een op maat gemaakte omgevingen die de problemen uit de weg gaan. Typische voorbeelden zijn (meerdere) dieptesensoren, roterende tafels of het gebruik van extreem veel cameras. Een dergelijke oplossing is duur wanneer hoge resoluties vereist zijn en zijn niet in staat om een volledige 3D-reconstructie in realtime te genereren (bv. dieptesensoren creëren slechts 2.5D-reconstructies en draaitafels kunnen alleen statische objecten reconstrueren). Bovendien is het soms onmogelijk (vanwege de kosten) of onpraktisch om op te schalen, bv. verkeersmonitoring, fabrieksbrede installaties, ... Die speciale sensoren laten toe om de 3D-structuur nauwkeuriger te reconstrueren met minder sensoren, maar ondervinden nog steeds problemen met occlusie. Tegelijkertijd kunnen camera's ook worden gebruikt voor fotorealistische reconstructies of om de omgeving beter te begrijpen.

We presenteren robuuste, krachtige algoritmen om nauwkeurige 3D-reconstructies mogelijk te maken in uitdagende omstandigheden met cameranetwerken waarbij de camera's deels dezelfde omgeving zien, maar

vanuit verschillende gezichtspunten. We bieden een oplossing voor de volledige pijplijn, van het capteren van beelden tot een fundamentele analyse van het 3D-model en 3D-pose-inschatting van mensen in de omgeving door 2D-poses te reconstrueren in 3D. Het inschatten van menselijke poses kan helpen om de prestaties van een atleet in sport te verbeteren of games te besturen zonder controllers. Sommige van deze applicaties vereisen realtime verwerking. Daarom wordt ook rekening gehouden met rekensnelheid in elke stap van de pijplijn.

We stellen een oplossing voor het occlusieprobleem door gebruik te maken van meerdere camera's die rond de persoon of het object zijn gepositioneerd. Door het aantal gezichtspunten te vergroten, verkleinen we de kans dat deze objecten tegelijkertijd in meerdere camera's worden geoccludeerd. Deze strategie stelde ons in staat een algoritme te ontwikkelen dat reconstructie kan uitvoeren terwijl occlusie kan worden afgehandeld door te redeneren op een geometrische niveau.

Een traditionele techniek genaamd *shape-from-silhouettes* benadert een 3D-reconstructie op basis van de silhouetten van dat object vanuit meerdere gezichtspunten en vormt de basis van dit werk. Omdat we voornamelijk met bewegende objecten te maken hebben, gebruiken we vaak voorgrond-/achtergrondsegmentatie om deze silhouetten te verkrijgen. Camerakalibratie is ook vereist omdat de relaties tussen camera's nauwkeurig gekend moet zijn om 2D-beeldcoördinaten om te zetten naar 3D-coördinaten. Om een object volumetrisch te reconstrueren, segmenteren we eerst de projectie ervan in elk camerabeeld.

Om menselijke bewegingen in detail te analyseren, presenteren we ook een raamwerk voor het schatten van lichaamshoudingen in 3D. Onze 3D-pose-inschatting maakt gebruik van de output van 2D-pose-inschatters, die gevoelig zijn aan verschillende fouten zoals het wisselen van ledematen en onbetrouwbare detecties. Onze bijdrage is de robuuste 3D-fusie van deze foutgevoelige geschatte 2D-poses. De resultaten laten zien dat de reconstructie voldoende dicht bij op commerciële markergebaseerde systemen ligt, die in dergelijke experimenten als de gouden standaard worden gebruikt.

De belangrijkste bijdragen van dit werk zijn:

- een real-time 3D volumetrische reconstructiemethode op basis van silhouetten;
- een real-time en betrouwbare 3D-pose-inschatter op basis van foutgevoelige 2D-poses;

- een robuust algoritme voor het afhandelen van occlusie voor het verbeteren van volumetrische reconstructie en 3D-pose-inschatting;
- de combinatie van volumetrische reconstructie en 3D-pose-inschatting om het begrijpen van de omgeving te bevorderen;
- een end-to-end realtime pijplijn van afbeeldingen/video's tot een volledige 3D-reconstructie.

Het onderzoek resulteerde tijdens dit doctoraat in 7 peer-gereviewde tijdschriftartikelen, waarvan één als eerste auteur. Bovendien werden 17 peer-gereviewde bijdragen gepresenteerd op internationale conferenties over computervisie, 7 als eerste auteur en 10 als co-auteur (zie Appendix A).

Abstract

When a camera captures its environment, the 3D structure is lost during projection on the image sensor. Computer vision deals with how computers can gain a high-level understanding from images or videos. Often, regaining a sense of these 3D structures is an essential step towards scene understanding. Observing the scene from multiple camera views helps to understand a scene better compared to a single image. By fusing the images from different camera views, it is possible to recreate the 3D shapes of the objects in the scene.

Applications include, for example, creating 3D models of real-world objects to measure specific properties such as volume distribution and circumferences. For instance, by reconstructing a plant at fixed time intervals, the plant's growth process can be monitored to analyse the impact of sunlight and fertiliser. Because computers become more and more powerful, we can even reconstruct 3D object for every set of images from different cameras. The volumetric information of such 3D reconstructions can be used for tracking people's movements in the scene or even for analysing traffic.

Augmented and virtual reality (AR/VR) form another application field for such a system because these type of applications depend heavily on the 3D real-world environment. Current solutions do not track the movement of the whole person, but only of specific devices worn or carried by the person, e.g., headsets and hand controllers. Full-body tracking can help to accomplish a more immersive user experience. Such an application requires volumetric shape reconstruction and real-time 3D human pose estimation.

In real-world environments, the cameras often cannot capture the entire objects/persons of interest due to obstacles between them and the camera. This phenomenon is called occlusion. Occlusions are always dependent on the camera's perspective. When an object is occluded in one view, it does not necessarily mean that it is occluded in the other

view. Occlusion mainly happens in three ways. Objects of interest move behind static obstacles in the scene (static occlusion), e.g., a person walking behind a tree, moving objects hide the object of interest temporarily (dynamic occlusion), e.g., a car crossing in front of a pedestrian, or parts of the object hide parts of the same object (self-occlusion), e.g., a person's arm in front of his torso.

State-of-the-art methods currently do not handle occlusion well. Typically, parts of the object that are not visible in one view are missing from the reconstruction, even when they are visible in some other views. Most of these 3D reconstruction methods assume that all cameras can observe all objects in the scene. For example, when a person is standing next to a table, the reconstruction may miss a leg because the table occludes the leg in at least one view. To avoid the impact occlusions can have on the 3D reconstruction, it is essential to have enough views left in which each part of the object is not occluded.

To overcome the challenges in reconstructing 3D geometry from 2D views, applications requiring high accuracy resort often to specialised hardware or highly tailored environments. Typical examples are (multiple) depth sensors, rotating tables or extremely dense camera networks. Such a solution is expensive when high resolutions are required or cannot offer a full 3D reconstruction in real-time (e.g.,depth sensors create 2.5D reconstructions, turntables can only reconstruct static objects). Moreover, it is sometimes impossible (due to cost) or impractical to scale up, e.g., traffic monitoring, factory-wide installations, etc. On the other hand, dedicated sensors allow one to reconstruct the 3D structure more accurately with fewer views but still suffer from occlusions. At the same time, general-purpose cameras can also be used for photorealistic reconstruction or to gain further understanding of the environment.

In this work, we present robust, high-performance algorithms to enable accurate 3D reconstruction in challenging conditions using a network of cameras with overlapping views. We offer a solution for the entire pipeline of the solution from input images to a fundamental analysis of the 3D model and 3D human pose estimation in the scene. We use 3D human pose estimation to assess an athlete's performance in sports, control immersive games and track workers in a factory. Some of these applications require real-time processing. Therefore computation speed is taken into account as well and essential in every part of the pipeline.

We propose a solution to the occlusion problem with multiple cameras positioned around the person/object of interest. By increasing the number of viewpoints, we reduce the chance that these objects are occluded

simultaneously in multiple cameras. Furthermore, this strategy allowed us to develop an algorithm that can perform 3D reconstructions while handling occlusion through geometric reasoning.

A technique called shape-from-silhouettes approximates a 3D reconstruction from the silhouettes of the object as seen from multiple viewpoints and forms the basis of this work. Since we mainly deal with moving objects, we often resort to foreground/background segmentation to obtain these silhouettes, but when real-time processing is not required, neural network segmentation often provides a reliable alternative. Camera calibration is also necessary because the relations between cameras are needed to map 2D image coordinates to the 3D world.

To further analyse human movement, we also present a 3D human pose estimation framework. The 3D human pose estimation uses 2D poses that are detected by 2D pose estimators, which are sensitive to several errors such as mislabelled limbs, misdetected keypoints and occluded keypoints. Our contribution is the robust 3D fusion of the often error-prone 2D poses. Furthermore, the results show that the reconstruction is close to marker-based systems, used as the gold standard in such experiments.

The main contributions of this work are:

- a real-time 3D volumetric shape reconstruction method based on silhouettes;
- a real-time and reliable 3D human pose estimation based on error-prone 2D poses;
- a robust occlusion handling algorithm that improves volumetric shape reconstruction and 3D human pose estimation;
- the combination of volumetric shape reconstruction and 3D human pose estimation to improve scene understanding;
- an end-to-end real-time pipeline from images/videos to a full 3D reconstruction.

The research during this PhD resulted in 7 peer-reviewed journal papers, with one as the first author. Moreover, 17 peer-reviewed conference papers were presented at international computer vision conferences, 7 as first author and 10 as co-author (see Appendix A).

List of Figures

List of Tables

Glossary

Camera calibration The mathematical representation which describes how a 3D scene is captured by a camera.

Keypoint A point detected by the 2D pose estimator. Each keypoint consists of a (x,y) coordinate and a confidence score. Most keypoints correspond to human joints.

Keypoint model The configuration in which the keypoints are connected to each other to form a model such as the OpenPose models MPI, COCO and BODY_25, both 2D and 3D.

Occlusion The phenomenon that a sensor cannot observe the object of interest due to objects positioned in between the sensor and the object of interest.

Pose A specific relative position in which a keypoint model appears in an image or in the 3D world.

Pose estimation The process of recognising parts of a person in an image to estimate how the body parts are relatively positioned toward each other.

Silhouette The projection of an object on the image sensor (with loss of texture). A silhouette in this work is usually represented as a binary image where the projection of the shape is white and the rest is black.

Shape reconstruction The process of generating a volume in 3D based on 2D silhouettes from different camera calibrated camera views.

Voxel An elemental cuboid of fixed dimensions (width, length, height), more or less the equivalent of a pixel in an image.

Chapter 1

Introduction

1.1 Problem statement

Professional athletes strive to achieve the very best in their sport. When sport-specific skills reach high levels, the margin to improve becomes slimmer. Moreover, to maintain a competitive edge over opponents, details are essential such as subtle differences in technique, which, for instance, allow more efficient conversion of energy from the muscles to the desired goal: faster, higher, stronger. Athletes with an efficient running technique run faster and keep this up for longer. At the same time, a javelin thrower benefits from an optimal throwing angle and transfer from muscle strength to the speed of the javelin. Each sport has its essential techniques, and coaches try to adapt an athlete's technique to find that edge over their opponents.

Nowadays, coaches often film their athletes to analyse their technique. Hudl Technique [44], and Coach's Eye [96] are mobile apps that facilitate video analysis during training. Despite their ease of use, it is often hard to understand the 3D body movement on the basis of a series of 2D images taken from the same viewpoint due to ambiguity of a pose and possibly invisible parts of the body, as seen from the particular camera perspective. Moreover, these apps do not provide automatic pose estimation to aid the trainer in understanding the motion in consecutive frames. The apps only provide a manual annotation tool to draw simple shapes on frames separate to illustrate technical aspects as feedback for their athletes. In Figure 1.1 the left and right legs are indicated using the manual annotation tool to assess the skater's position at the entry of the bend. One can quickly scroll to the previous and next frames of a video using the controls at the bottom of the screen, but the annotations

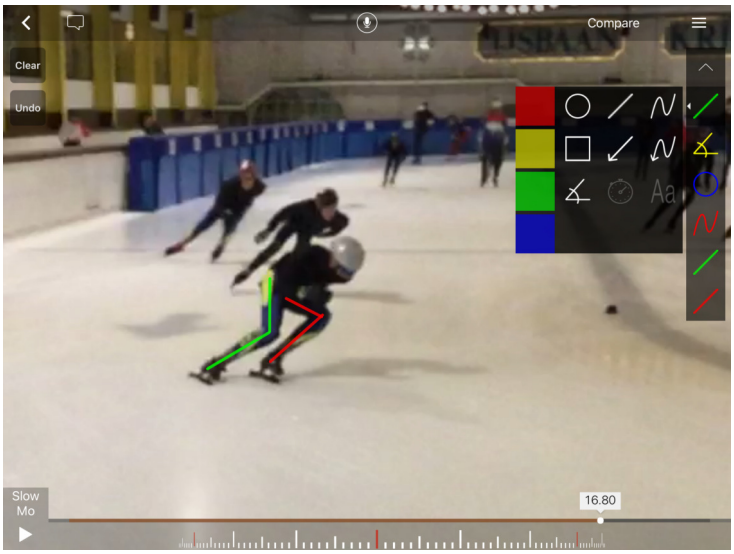


Figure 1.1: *Hudl Technique: indicated lines are manually annotated to asses skating technique at the entry of the bend.*

are not updated automatically. Arm positions are also challenging to analyse from this perspective due to the similar colour of the upper body clothing and invisibility of the left arm. A number of building blocks in this work can be used to automate the annotation process and provide more objective assessments of the human movement.

To fully understand the scene in which an athlete executes a particular technical exercise, 3D reconstructions of the athlete and attributes such as a badminton racket, a javelin or a ball are required. Single-camera 3D reconstruction remains a challenging problem due to the loss of depth information after projection. Therefore, 3D positions are either obtained in combination with depth sensors or by using multiple cameras. In general, depth cameras have a limited range and smaller image resolutions than visible-light cameras. Moreover, depth cameras do not produce a complete 3D reconstruction from a single point of view. Multiple cameras which are positioned around, accurately track the 3D motion of objects and humans. Several joints can be accurately tracked using single-view pose estimators on images from different viewpoints and reconstructing a 3D pose of the athlete. Such a pose is helpful to assess biomechanical motion performed by the athlete (Figure 1.2).

A keypoint model is well defined for humans but it is limited to information about the relative positions of body parts, which is very valuable, but only tells part of the story. At the same time, such a keypoint

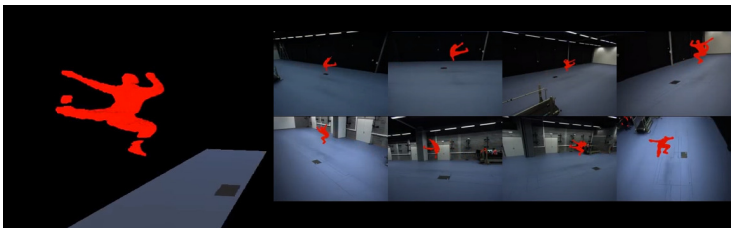


Figure 1.2: *Sport Science Lab Jacques Rogge: Shape reconstruction of an air kick. The athlete is segmented in each image and reconstructed in 3D.*

model does not consider volumetric information. Volumetric information defines the space occupied by a person in the scene more accurately than a keypoint model. It is therefore better to use it to assess whether or not an athlete would be able to fit in between a gap created by other players, for instance. Moreover, most rigid objects do not necessarily contain articulated joints of a keypoint model, e.g., badminton rackets or balls. In these cases, volumetric 3D reconstruction offers an alternative way to analyse the motion of objects in general without the need for a skeleton representation. The combination of pose reconstruction and volumetric reconstruction creates a richer context to analyse the object's movement as each of these reconstructions have complementary benefits and drawbacks.

A typical problem when working with images and videos is occlusion. Occlusion is the phenomenon that occurs naturally when a sensor cannot entirely observe its target within its sensing area. This phenomenon occurs when an object is positioned in front of another object. Therefore, the camera cannot observe the entire object behind the occluding object. Occlusion manifests itself in different ways. The occluding object is either moving in the scene (dynamic occlusion, e.g., a ball in front of a player) or it remains at the same position (static occlusion, e.g., the pole of the badminton). Another important distinction is between object-to-object occlusion, e.g., a player in front of another player, or self-occlusion, a part of an object is occluding another part of the same object, e.g., the crossed arms in front of a person's upper body.

Occlusion increases complexity when assessing the pose and shape of an object/person due to incomplete and incorrect data from the sensors. The research reported in this thesis will show how multiple camera observations may improve our understanding of the pose and shape of such an occluded object by using geometric reasoning between camera views. Therefore, the proposed method shows why a camera network with overlapping views is beneficial for occlusion reasoning and is capable

of creating more reliable 3D reconstructions.

1.2 Background and related work

Other researchers have investigated solutions for the problem statement. This section provides an overview of the state-of-the-art methods and available alternatives to the proposed solution. Since shape reconstruction and 3D human pose estimation are two different problems that may require different technologies, we discuss them separately.

1.2.1 Volumetric shape reconstruction

The surface of an object needs to be estimated to obtain volumetric shape reconstruction. 3D information can either be sensed directly by using depth sensors (e.g., stereo cameras as in [48, 101, 113, 94], time-of-flight cameras [24], or micro-lens cameras [103]) or by observing 2D silhouettes of the object from multiple viewpoints with general-purpose RGB cameras.

Depth sensors may seem like the most logical choice as they provide 3D points directly. However, these sensors usually provide lower image resolution, have limited range, and only measure so-called 2.5D because they can only measure the closest point from the object to the sensor. Avoiding this requires multiple depth sensors, but these sensors may interfere with each other. Depth sensors use short light pulses to measure the time-of-flight or structured light where a known pattern is projected onto the object (either in visible light or infrared light). The distortion of the pattern translates to a depth image. However, such specialised sensors are not as widely adopted as general-purpose cameras, which are already ubiquitous in many sites, including industrial environments and traffic intersections. The depth measurements contain inaccuracies due to noise, and the resolution of these sensors is still limited.

General-purpose cameras are widely available within an extensive range of specifications. Image resolutions and maximum framerate can be chosen depending on the use case. Such cameras are also non-intrusive because they do not interfere with the scene or each other. In general, visible light cameras can see further than depth cameras. The depth is obtained using camera calibration in combination with object detections (silhouettes).

Shape reconstruction from images of an object taken from different viewpoints can be used to estimate the volumetric reconstruction of that

object based on the backprojection of the silhouettes, as proposed by the shape-from-silhouettes (visual hull) concept by Laurentini in 1994 [52]. The reconstructed shape is the intersection of the generalised cones, each defined by the camera position and the silhouette from each corresponding camera view.

Incomplete silhouettes deteriorate the estimated shape because shape-from-silhouette algorithms assume perfect silhouettes. Three types of errors are common: inaccurate silhouette boundaries, holes in the silhouettes, and parts of the silhouette are missing. Such incomplete silhouettes may be due to errors in the segmentation algorithm, such as foreground/background (FG/BG) segmentation, but their primary cause is occlusion. If a static object is positioned between the camera and the moving object, foreground/background segmentation cannot segment parts of the silhouette of the moving object. However, occlusion is very common in many real-life situations. In indoor as well as outdoor environments, occlusion is often inevitable.

Occlusion handling has been studied before, and the existing approaches can be divided based on the prior knowledge and the number of sensors. Xiang et al. [105] proposed a single-camera approach to detect multiple occluded objects. They used 3D perspective to handle various occlusion patterns between objects. Partial observations of the known objects are used to determine their position and orientation. However, such methods can only be applied to rigid objects with a known appearance model.

Occlusion-proof 3D reconstruction methods use appearance models of the object of interest and use part-based detection to recognise the partly visible objects in the scene [28, 31, 61, 72, 73]. While these methods work well for the reconstruction of known objects, they can only be generalised to a broader range of objects if a dataset is available to model all these different objects. Detecting all different possible objects will also considerably slow down the reconstruction process.

The more general approaches do not use prior knowledge about the objects in the scene. Favaro et al. proposed a method that considers occluded regions in an image as blind spots in the camera view, even if the occlusion was only temporarily [30]. Also, this method does not care whether an object appears in front of an occluder.

Extended visual hulls are extensions of the standard shape-from-silhouettes algorithm. Guan et al. proposed this concept in [37]. The paper shows that extending silhouettes into regions suspected of occluding yields more accurate 3D reconstruction than ignoring the occluded

areas completely, as shown by Favaro [30]. However, occluded regions are still treated as utterly devoid of information even when objects pass in front of the occluding object. Díaz-Más et al. [22, 23] and Landabasso et al. [51] relaxed the reconstruction process in such a way that one or more of the cameras may have unreliable silhouette information, without explicitly modelling in which cameras or in which parts of the scene this occurs. Such an approach gives rise to reconstructed volumes that are typically too large but include the original object. In [38], Haro et al. extended this approach to estimate the number of unreliable cameras locally rather than globally, using an a priori object model to reduce the overestimation of the volume.

The shape reconstruction method proposed in this dissertation is similar to extended visual hull methods because it also aims to reconstruct the shape of arbitrary objects of unknown size and complexity in a way that maximally agrees with the incomplete silhouettes. However, our approach is computationally less demanding. We also create a model to localise occluders in the scene. We achieve lower complexity by reasoning on regions of connected voxels rather than individual voxels.

1.2.2 3D human pose estimation

Separately from the reconstruction of arbitrary objects, the reconstruction of specific key points of an object is also a valuable asset in scene understanding. 3D positions of keypoints of the human body lead to 3D human pose estimations that can be used to understand a person's behaviour in the scene. The 3D pose can determine which action a person is performing and serves human motion analysis.

Current experiments in the field of human motion analysis are often performed with marker-based systems such as Qualisys [81], Vicon [65], and OptiTrack [67]. The biggest drawback of these systems is the time needed to equip a person with reflective markers. Moreover, markers may fall off during the data capturing process, rendering that recording useless. Besides this, such a system with markers cannot be deployed in numerous applications such as virtual classrooms or athlete analysis during competition.

Recent advances in markerless monocular pose detection enable new applications that require semi-accurate tracking of body parts. Such markerless systems provide the solution for the drawbacks of marker-based systems mentioned above. Whereas marker-based systems claim submillimeter accuracy for the markers, markerless systems only obtain

an accuracy up to a few centimetres. The reason is that a pose estimator does not always detect a keypoint (e.g., an ankle) at the anatomically correct position. For optimal results, it is best if the subject wears tight clothing to avoid that keypoints might not be visible. Even humans would have a hard time locating the exact position of the joints from the videos only.

Motion analysis often makes use of the changes in planar joint angles, e.g., technical performance in sports or basic clinical gait analysis. For this reason, a markerless system has its value even though it cannot accurately measure rotations about the limbs axis. Markerless systems have been around for a while now. Since the early 2000s, research has been ongoing to locate keypoints in RGB videos. Most of these approaches relied on shape-from-silhouettes and tried to match a detailed kinematic model. Positional errors were typically larger than 100 mm [19, 18, 78, 79]. Later advances obtained 50 to 100 mm positional errors on the joints [43, 45]. More recently, the shift to monocular pose extraction enabled more flexible camera setups [27, 29]. However, the reported positional errors of these systems are typically between 50 and 150 mm.

Additionally to obtaining better accuracy, we also aim to improve robustness. Unlike the markerless human pose estimation methods mentioned before, our solution uses the existing 2D pose estimator of OpenPose [16] and triangulates the pose in 3D. Although these pose estimators produce impressive results, they are not perfect. Typical problems related to the use of a single camera are undetected keypoints, misdetected keypoints due to occlusion of view-related ambiguity, and mislabeled keypoints e.g., swapping of the left and right leg. We handle all three issues in this work by fusing the information from multiple viewpoints and present a robust system for a wide range of applications because of its flexibility in the number of cameras and scalability. The proposed system can accurately detect keypoints with positional errors between 25 and 50 mm. Such accuracy is lower than the state-of-the-art methods but is accurate enough for numerous applications such as game controlling, behaviour analysis and technical analysis in sports. The accuracy could be further improved when the camera calibration becomes more accurate, the camera resolution increases and the pose estimators detect keypoints more reliable.

1.3 Applications

A network of cameras observes the scene from different viewpoints with overlapping views to enable 3D reconstruction. On the one hand, we estimate 3D poses to better understand human movement and behaviour. On the other hand, we approximate volumetric shape reconstructions of persons and objects. The combination of 3D human pose estimation and volumetric shape reconstruction has numerous applications outside sports, such as in entertainment, for rehabilitation, and in manufacturing industry. Depending on the applications, the required precision differs between a few millimetre (AR/VR) and a few centimetres (sports, immersive gaming, rehabilitation, manufacturing).

The entertainment business, for instance, may benefit from detailed 3D human pose estimation. VR applications today become even more immersive for players if they can see the movements of their own body through the head-mounted display (HMD). Nowadays, VR systems only track specific devices worn by the player, such as the HMD and hand controllers. Therefore, these systems ignore certain body parts, such as the legs, which breaks the immersiveness of such a system. Another application in the entertainment business are games with freedom of movement. These games are not restricted to a limited number of motions (e.g., a game controller with a few buttons) but interact with the player more naturally and without a controller/wearable. The proposed system is similar to the Xbox Kinect, but our system is not limited to actions in front of one particular camera.

The work can also be used for the purpose of rehabilitation. Numerous people need to rehabilitate after an accident or an injury. Part of rehabilitation is the correct execution of specific exercises defined by a physiotherapist. A physiotherapist cannot watch each patient executing every single exercise. Moreover, the patient usually practices without supervision. However, when a patient executes an exercise in the wrong way, it may lead to new injuries or the straining of weak muscles and joints, which prolongs the rehabilitation period. Therefore, a monitoring system that automatically detects if exercises are performed correctly has its merits.

Camera surveillance often uses a camera network, but to reduce the cost these cameras rarely have overlapping views. In certain circumstances, however, it is worth investing in a camera setup with overlapping views, for instance, to reliably determine if there are multiple people in a space specifically designed for a single person, e.g., a bank vault to monitor for possible threats and security breaches. The surveillance system becomes

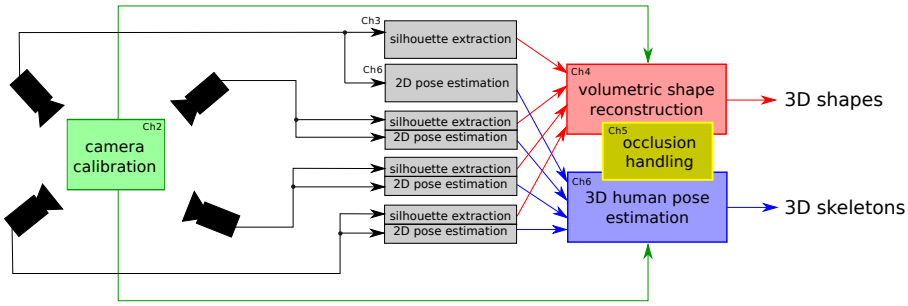


Figure 1.3: *System overview: images are captured from cameras and two separate reconstructions are calculated: 3D shapes and 3D poses which each handle occlusions (chapters in this book indicated with ChX).*

much more robust by using multiple cameras since the information from different viewpoints can be aggregated.

Monitoring workers in a factory can help to optimise specific processes. For instance, common parts needed for an assembly process on a conveyor belt can be placed at easily reachable positions to avoid straining the worker's back. By understanding the behaviour of a worker and the manipulation of objects, it is also possible to semi-automatically generate a list of work instructions from an expert recording and hence avoid the tedious task of generating these instructions. Factories use these instructions to teach other workers to execute the same process.

1.4 System overview and outline

In this section, we explain the system and its challenges under various real-world circumstances. Figure 1.3 above shows an overview of the essential parts of the system and refers to the chapters in which we explain the building blocks in more detail. On the left, we see several cameras. Each of these cameras captures images that are processed in two ways: silhouette extraction and 2D pose estimation. The volumetric shape reconstruction (red) processes the resulting silhouettes, and the 3D human pose estimation processes the resulting poses (blue). Both reconstructions benefit from occlusion handling (yellow). For each set of frames from multiple cameras, we produce 3D shapes and 3D poses.

The camera setup needs to be calibrated, which means that we need to know the position, orientation and intrinsic parameters (focal length, centre of projection, …) of each camera. Both reconstruction processes rely on camera calibration. In theory, we need to calibrate once for a

fixed camera setup. However, temperature changes and external forces may cause inconsistencies in the camera calibration and require recalibration. Chapter 2 reviews multi-camera calibration and overviews common camera parameters and their functions. It also guides us on how to perform camera calibration for multiple cameras in a wide-baseline setup.

For the volumetric reconstruction, the pixels belonging to people or objects must be segmented in each image from different viewpoints. In Chapter 3 we discuss how we extract silhouettes of the objects and people using segmentation. We distinguish between colour segmentation and foreground/background segmentation. The latter is often most convenient since the focus is on moving objects/persons primarily. It is often best suited in real-world environments due to the lack of distinctive colours between background and foreground.

The silhouettes are backprojected in the 3D space using the camera calibration. The backprojected silhouettes form generalised cones with the camera position as their tops. The intersection of all these cones represents an approximation of the person or object in 3D. In general, the more views there are, the more precise the reconstruction becomes. 3D reconstruction is discussed in Chapter 4. We discuss different representations of a 3D model and explain 3D reconstruction from a set of calibrated camera images using silhouettes for the visual hull and additional features such as colour for the photo hull.

Assuming no occlusion and perfect segmentation and detection, the result of this system is a volumetric shape reconstruction which encapsulates the 3D pose. However, in real-world environments, occlusion and segmentation errors will introduce errors and create challenges to be tackled. Occlusion occurs when a camera cannot observe the entire object of interest due to obstacles between the camera and the object of interest. Different types of occlusion may occur: object-to-object occlusion or self-occlusion. In the latter case, the object itself occludes another part of the same object, e.g., an arm in front of a person's body. Occlusion, if handled poorly, leads to incomplete silhouettes and incomplete 3D reconstruction. A similar problem arises when we cannot segment an object due to similar foreground and background pixels. Chapter 5 presents our approach to detect and handle occlusion in detail. We explain geometric reasoning and show how to reconstruct an object in 3D in the presence of occlusion.

For 3D human pose reconstruction, we match 2D poses in different viewpoints and triangulate these corresponding 2D poses to estimate the

3D pose while accounting for labelling and positional errors. Such pose estimators need large amounts of training data and are object-specific. In this investigation, we solely focus on human pose estimation, but we could extend the same technique to other reliable non-human pose detectors. A solution is presented for the cross-view pose correspondence problem to reconstruct multiple people in the same scene. We discuss the entire process in Chapter 6.

While reconstructing multiple objects simultaneously, another challenge arises. Both volumetric and pose reconstruction underperform when 2D silhouettes or 2D poses are matched incorrectly, leading to ghost shapes that are not present and incorrect 3D points. 2D and 3D tracking of the persons/objects partly solves this issue, but with limited camera views and an increasing number of people, the challenge persists. Therefore, it is essential to realise that the number of cameras required for a particular use case depends on the reconstruction volume and the scene's complexity. More people generally means more required cameras and this should be carefully analysed in detail for each case.

The number of cameras and camera specifications also has impact on two other challenges: precision and computation speed. Depending on the application, the results need to be precise (for instance, subtle motion of the body during a physiological exercise) or computed in real-time. In a live system, the cameras send their images to the computer in real-time, which imposes a hard requirement on the bandwidth (usually using a gigabit ethernet network). Image resolution, frame rate and the number of cameras determine the required bandwidth. The bitstream should never reach the maximum available bandwidth to avoid frame drops as much as possible.

In Chapter 7 several applications are discussed in which our 3D reconstruction algorithm has proven its usefulness. Examples are the static reconstruction of objects, the dynamic reconstruction of sports persons and workers. Each use case has its challenges.

We conclude this dissertation in Chapter 8 with a summary of the obtained results.

1.5 Contributions and publications

Several research projects and collaborations with companies have made use of the work reported in this doctoral dissertation. The main projects were an FWO project, iRUNman, StarTT Complexity, imec.icon IPLAY and imec.icon COSMO.

- An FWO project led to a multi-camera dataset, recorded in a holiday home in Mol. We evaluated several trackers on this challenging dataset with limited overlap between camera views and real-world occluders. Our detector based on shape-from-silhouettes reconstruction produced the best results for all individual trackers investigated. This resulted in two publications: a conference paper [85] at ICDSC 2015 and a journal publication [68].

- The iRUNman project is a prestigious project that led to the Sports Science Lab Jacques Rogge opening. In the biomechanics lab, we installed a camera setup of 8 fixed cameras controlled from two capture PCs and with an ability to record at 67 fps at a resolution of 780x580. We used this system to capture people while walking, running, playing basketball and playing badminton for research in gait analysis and technical analysis in sports in co-operation with the Department of Movement and Sport Sciences of Ghent University.

- The StarTT project Complexity was a valorisation project for manufacturing industry where we monitored workers at a conveyor belt. We used five cameras in this project to capture the worker and detect specific events, such as picking items from a large storage rack. The trajectories of the person were captured and used for anomaly detection and clustering. We were able to extract how many times a certain pickup happened and if there were any anomalies from this data. This research led to four conference papers ([4], [108], [107] and [5]) and one journal paper [6].

- The imec.icon IPLAY project resulted in a fully integrated entertainment unit with pressure sensors, LEDs in the floor and a wall projection. We added four cameras to this system to track the 3D pose of the player in order to manipulate games. There were two applications in this project: gaming and rehabilitation. At the end of the project, the demonstrator showcased that all components in the system worked nicely together. The work in this project led to two conference papers ([2] and [90]) and one journal paper ([36]).

- The imec.icon COSMO project investigates the semi-automatic generation of work instructions for line work. An expert demonstrates a series of instructions captured by cameras and microphones. We track the worker and extract events from his movements to identify the instructions. Voice recognition in co-operation

with object detection on the video enriches the instruction information.

In total, the work reported in this dissertation contributed to the publication of 7 peer-reviewed international journal articles, of which one as a first author, and 17 international conference papers, 7 of which as the first author. For a complete list of publications, we refer to Appendix A. Aspects of this work were demonstrated at several notable events such as iMinds the Conference 2016, Sports Innovation Congress 2017 and ITF Future Summits in 2019. We also participated in a valorisation trajectory during the iMinds iBoot 2015. See Appendix B for more information about these events.

Chapter 2

Camera Model and Multi-camera Calibration

We reconstruct 3D models based on 2D images captured from different viewpoints in this work. It is crucial to know where these cameras are positioned and how they are rotated in relation to each other and the world coordinate system. More specifically, a camera captures a 2D projection of the 3D scene with a loss of implicit depth information. In order to attempt to reverse the projection process (backprojection), we need to understand precisely how images are formed on the sensor.

The imaging process of a camera can be defined by two sets of parameters: the intrinsic parameters and the extrinsic parameters. Intrinsic parameters are those that are specific to the internal workings of the camera. These parameters describe how the light travels inside the camera (from lens to image sensor). The extrinsic parameters describe the position and orientation of a camera in the 3D space to a world coordinate system.

Camera calibration represents the process of estimating the camera parameters mentioned above. In that way, it is possible to relate all cameras to a shared world coordinate system so that pixels from different camera images can be used to reconstruct the 3D scene. Therefore, camera calibration is an essential step towards 3D reconstruction from multiple viewpoints. Errors in the calibration lead to inferior 3D reconstruction. Significant errors may even render the reconstruction useless.

This chapter is structured as follows. Section 2.1 explains a camera model which is widely used. We explain the different intrinsic and extrinsic parameters of this model and how calibration processes are used

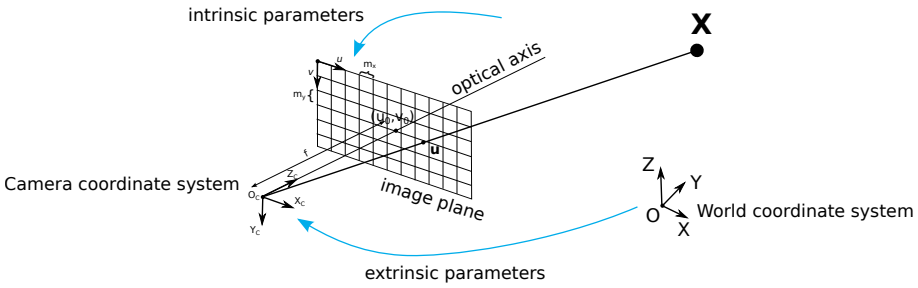


Figure 2.1: *The pinhole camera model. The perspective transformation of 3D point* $\boldsymbol{X}$ *in the world coordinate system to the 2D point* $\boldsymbol{u}$ *on the image plane depending on the intrinsic and extrinsic camera parameters.*

to estimate them using methods from literature. We also demonstrate how camera coverage maps can be used as a tool to visualise which part of the scene is covered by how many cameras in Section 2.3. This tool can help decide where additional cameras should be positioned to increase camera coverage in certain areas. In Section 2.4 we perform several experiments to show the impact of common errors in the camera calibration on the image formation process. Finally, our conclusions about camera calibration are presented in Section 2.5.

2.1 Camera model

A broadly used model to represent a camera mathematically is the pinhole camera model that has been around from before the 20th century. In this model, a scene view is formed by transforming 3D points with homogeneous coordinates $\boldsymbol{X} = \begin{bmatrix} X & Y & Z & 1 \end{bmatrix}^T$ in the world coordinate system to homogeneous 2D image points $\boldsymbol{u} = \begin{bmatrix} su & sv & s \end{bmatrix}^T$ using a perspective transformation. The perspective projection can be represented as a matrix P, so that we can express the mapping of $\boldsymbol{X}$ to $\boldsymbol{u}$ as

$$\boldsymbol{u} = P\boldsymbol{X}. \tag{2.1}$$

The matrix P decomposes as

$$P = A[R|\boldsymbol{t}], \tag{2.2}$$

in which A is the camera matrix related to the intrinsic camera parameters and $[R|\boldsymbol{t}]$ defines transformation between the world coordinate system $\boldsymbol{O}XYZ$ and the camera coordinate system $\boldsymbol{O_C}X_CY_CZ_C$, which is related to the extrinsic camera parameters. Figure 2.1 visualises

this projection process. The following section will discuss the intrinsic parameters and extrinsic parameters in more detail.

2.1.1 Intrinsic parameters

I. Camera matrix

The intrinsic camera parameters fit nicely together in the camera matrix A and describes how to transform 3D coordinates to pixel units:

$$A = \begin{bmatrix} \alpha_x & \sigma & u_0 \\ 0 & \alpha_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{2.3}$$

This camera matrix contains five parameters. The parameters $\alpha_x = fm_x$ and $\alpha_y = fm_y$ represent focal length in terms of pixels, where m_x and m_y are the scale factors relating pixels to distance and f is the focal length. The principle point $\begin{bmatrix} u_0 & v_0 \end{bmatrix}^T$ tends to be in the middle of the image sensor in case the lens is mounted centrally and parallel to the image plane. The principle point corresponds to the point where the optical axis (Z_C) intersects with the image plane. The parameter σ represents the skew coefficient, which is non-zero if the sensor is not parallel with the imaging plane.

Real cameras use lenses to capture enough light in a short amount of time. The choice of a lens depends on the desired field of view. Action cameras often have fisheye lenses to capture more of the surroundings, while telescopic lenses have a narrow field of view to capture an image from a far distance. Lenses make the imaging process a bit more complex due to possible lens distortion. Therefore, this distortion needs to be modelled to find where a 3D point projects on the image sensor. In Figure 2.2 we see how a converging lens refracts incoming light and forms an image of the red object on the image plane. Note that multiple light beams starting from the same point end up in the same place on the image plane. This figure also illustrates how a lens helps to focus more light on one pixel than a pinhole camera.

II. Lens distortion

Since most lenses are not perfect, the image will not perfectly correspond to a perspective projection of the 3D scene because lens distortion will add a non-linear transformation to the imaging process. In Figure 2.3

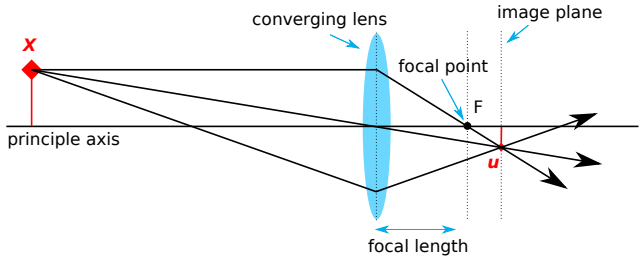


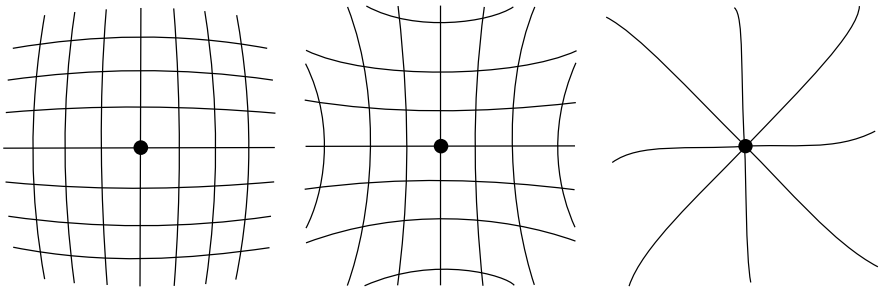
Figure 2.2: *Light refraction in a converging lens. The light rays reflecting on* $\boldsymbol{X}$ *are focused on the point* $\boldsymbol{u}$ *on the image plane. Light rays perpendicular to the lens (as if they originate from infinity) are refracted by the lens in a way that they pass through the focal point of the lens. Light rays having different direction, but also reflecting at* $\boldsymbol{X}$ *are all focused on* $\boldsymbol{u}$ *because of the converging lens.*

(a) *Original image* ***(b)*** *Undistorted image*

Figure 2.3: *Undistortion process on an image from the traffic intersection dataset in Ghent. In (a), straight lines do not appear straight due to lens distortion. Therefore, points A, B and C are not collinear. After the undistortion process, the three points are collinear in (b).*

the red line in the left image is a straight line in 3D. However, due to lens distortion, it appears as if it is curved. In order to relate pixels to the 3D world, this phenomenon needs to be modelled. The Brown-Conrady model is often used to model distortion [13] so that an image can be undistorted. In the right image, we see the result of this undistortion process. The curved lines convert to straight lines.

Two types of distortion are handled in the model: radial distortion and tangential distortion. Examples of radial distortion are shown in Figure 2.4a (barrel distortion) and 2.4b (pincushion distortion). In the case of barrel distortion, image magnification decreases with distance from the optical axis (usually near the centre of the image). The apparent effect is that of an image that has been mapped around a sphere (or barrel). In the case of pincushion distortion, image magnification increases with the



(a) Barrel *(b) Pincushion* *(c) Tangential*

Figure 2.4: *Different types of radial and tangential distortion. The black dot represents the optical centre. All lines in the images represent how straight lines which are parallel to the image sensor are distorted.*

distance from the optical axis. Lines that do not pass through the optical centre of the images are bowed inwards, like a pincushion. Tangential distortion occurs when the image plane and the lens are not parallel (lens misalignment). The effect of tangential distortion is shown in Figure 2.4c. We see that the lines become more curved further away from the optical centre.

In the remainder of this section, we explain the mathematics to undistort a point based on the Brown-Conrady model, after which we briefly discuss the calibration process. We consider five distortion coefficients: k_1, k_2 and k_3 to model radial distortion and p_1 and p_2 to model tangential distortion, which suffices for common camera lenses.

To convert a point captured by a camera to its corresponding undistorted position, we first normalize the image coordinates because all distortions discussed above are functions of the distance to the optical centre. The point $\begin{bmatrix} u & v \end{bmatrix}^T$ represents the undistorted point and $\begin{bmatrix} u' & v' \end{bmatrix}^T$ the corresponding normalized point:

$$\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} \frac{u-u_0}{\alpha_x} \\ \frac{v-v_0}{\alpha_y} \end{bmatrix}.$$

The most common form of lens distortion is radial distortion. Radial distortion occurs when light rays bend more near the edges of a lens than they do at the optical centre of the lens. Radial distortion converts points $\begin{bmatrix} u' & v' \end{bmatrix}^T$ to $\begin{bmatrix} u'_r & v'_r \end{bmatrix}^T$:

$$\begin{bmatrix} u'_r \\ v'_r \end{bmatrix} = (1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \dots) \begin{bmatrix} u' \\ v' \end{bmatrix}, \tag{2.4}$$

where $r = \sqrt{u'^2 + v'^2}$. If $k_1 > 0$, the lens has positive radial distortion (barrel distortion). If $k_1 < 0$, it has negative radial distortion (pincushion distortion). 3 parameters usually suffice to model the radial distortion accurately.

Tangential distortion is modelled with two distortion coefficients: p_1 and p_2. Using the same notation as with radial distortion we may express pure tangential distortion as follows:

$$\begin{bmatrix} u'_t \\ v'_t \end{bmatrix} = \begin{bmatrix} u' + 2p_1u'v' + p_2(r^2 + 2u'^2) \\ v' + 2p_2u'v' + p_1(r^2 + 2v'^2) \end{bmatrix}. \tag{2.5}$$

III. Intrinsic camera calibration

The estimation of the intrinsic parameters is usually performed by waving a planar surface with known feature points around in front of each camera. The camera calibration parameters cannot be estimated correctly from a single image due to the large number of parameters that need to be estimated. Completely different cameras and lenses can generate the same image. For example, the image of a calibration board captured by a zoom lens from far away may look the same as an image of a calibration board close to the camera with a lens having a short focal length. Therefore, multiple positions and orientations of the same board are required to estimate the set of camera calibration parameters for the used camera. A sufficient number of sufficiently independent observations is needed to estimate all camera parameters discussed in the previous section and properly constrain the optimisation to yield a well-defined answer. Otherwise, the routine may converge to some (local) minimum.

Checkerboard patterns are often used as planar calibration patterns because simple computer vision algorithms accurately and automatically detect the inner corners of such a pattern. Figure 2.5 shows that for a $M \times N$ checkerboard, there are $(M-1)(N-1)$ of such corner points detected. The coloured horizontal lines in the image belong to the same camera row on the checkerboard. These lines appear curved, which points to radial distortion.

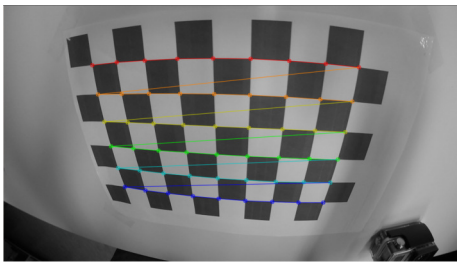


Figure 2.5: *Recognition of the checkerboard pattern. Each inner corner of the checkerboard pattern is detected automatically. The lines represent predefined connections between the feature points. These detected points are used to calibrate a camera intrinsically. Typically, between 20 and 50 images with differently oriented checkerboard patterns are required for a reliable calibration.*

The general intrinsic camera calibration algorithm consists of multiple steps, and most implementations are based on the papers of [42] and [112]. The calibration process is summarised in [98] as follows:

1. Print a pattern and attach it to a planar surface.
2. Take a few images of the model plane under different orientations by moving either the plane or the camera. Typically, between 20 and 50 checkerboard frames are used for calibration.
3. Detect the feature points in the images (e.g., checkerboard corners);
4. Estimate the five intrinsic parameters and all the extrinsic parameters using the closed-form solution as described in [11];
5. Estimate the distortion coefficients on the image using Equations 2.4 and 2.5
6. Calculate the m homographies between the known checkerboard layout and its projections in the n images (using least-squares on the overdetermined system).
7. Calculate the expected position of all m points in each of the n images using the estimated homography and distortion coefficients.
8. Calculate the total reprojection error as the sum of the distances between all $m \times n$ observed points and their expected position.
9. Iteratively refine the distortion coefficients and intrinsic parameters using the Levenberg-Marquardt algorithm to minimise the total reprojection error (steps 5-8).

In this work, we have created a calibration tool in which the calibration process is fully automated for the user from step 3 onwards. As such, the intrinsic calibration of a single camera takes less than 30 seconds.

2.1.2 Extrinsic parameters

This section will discuss the transformation matrix, which contains the extrinsic parameters. We also discuss how these parameters can be estimated using the extrinsic camera calibration methods. Extrinsic camera calibration is an essential step towards 3D reconstruction from multiple cameras because it defines the transformation from the world coordinate system to the camera coordinate systems. It is possible to relate 3D points to 2D image points from different cameras. The reverse process, called backprojecton, is when a pixel maps to a 3D line on which the corresponding 3D point is located. Backprojection forms the basis for 3D reconstruction from multiple cameras and will be discussed in more detail in Chapter 4.

I. Transformation matrix

The transformation matrix, which maps world coordinates ($\boldsymbol{X}_w$) onto camera coordinates ($\boldsymbol{X}_c$), consists of two parts: the rotation matrix and the translation vector. The rotation matrix R is an orthogonal matrix which expresses how the camera is rotated with respect to the world coordinate system. The translation vector $\boldsymbol{t}$ expresses the offset from the origin of the world coordinate system to the origin of the camera coordinate system. Therefore, the transformation matrix $[R|\boldsymbol{t}]$ expresses how the world coordinate system transforms to the camera coordinate system:

$$\boldsymbol{X}_c = [R|\boldsymbol{t}]\boldsymbol{X}_w \tag{2.6}$$

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \\ 1 \end{bmatrix}. \tag{2.7}$$

The transformation from one coordinate system to another is well defined in mathematics. The rotations can be expressed as matrix multiplications of rotations around a number of the main axes X, Y or Z, respectively θ_x, θ_y and θ_z. Figure 2.6 illustrates how the transformation matrix can be decomposed into four different parts: a translation of the coordinate system and three rotations: roll, pitch and yaw. The order of rotations and the choice of which axis to rotate around can be chosen.

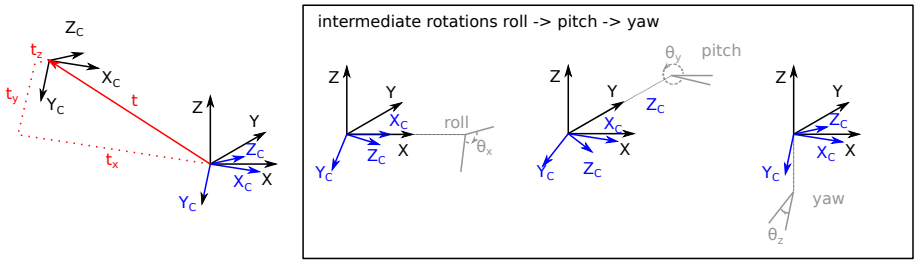


Figure 2.6: *Transforming between the world coordinate system and the camera coordinate system. The separate rotations are shown by roll, pitch and yaw. The translation is represented vector* $\boldsymbol{t} = (t_x, t_y, t_z)$.

II. Extrinsic camera calibration

The extrinsic camera parameters can be estimated using an extrinsic camera calibration process. We consider methods that can be used to calibrate multiple cameras simultaneously, e.g., stereo or point calibration and methods where each camera is separately calibrated in relation to a shared world coordinate system, e.g., using a 3D calibration object.

Often, extrinsic camera calibration is, just like intrinsic calibration, performed utilising a planar calibration pattern. For stereo cameras, it is convenient to wave a planar calibration pattern in front of both cameras at the same time [12]. However, when the cameras are further apart, e.g., wide-baseline camera setups, it is no longer possible or practical to orientate the planar calibration patterns so that the feature points are visible by both cameras, let alone to find enough different independent board orientations. However, as long as there is enough pairwise overlap, cameras can be calibrated in pairs.

Alternatively, multi-camera calibration can also be performed by moving around one point that multiple cameras can easily track simultaneously. [95] proposed a method using a slightly adapted laser pointer so that the light shines in all directions and uses bundle adjustment to optimise the measurements. [34] on the other hand, presented calibration techniques based on spheres of which both the centre and surface were used to estimate the extrinsic camera parameters. Another method proposed in the same work was to use the centreline of a person to estimate the extrinsic camera parameters. However, all these calibration methods calibrate the cameras up to a scaling factor. An extra alignment step helps transform the calibration to a world coordinate system with physical units e.g., mm and place the origin of the coordinate system to the desired location. All methods mentioned in this paragraph have been explored during this

Figure 2.7: *Example of the POSIT algorithm on the traffic intersection dataset of Ghent. The centres of the red circles correspond with the annotated positions. The white circles indicate the position of these points using the estimated camera parameters. Ideally, the centres of both circles are the same.*

PhD but are often not practical to use if surrounding light cannot be controlled. Therefore, the detection of a point light source or spherical lamp becomes inaccurate.

A practical method requires minimal manual work to calibrate cameras by using a calibration object. Several keypoints are defined, of which the world coordinates are known. A minimum of 4 non-coplanar points are needed to calibrate one camera, and the image locations of these points are usually indicated manually. Since each camera is basically calibrated independently, recalibration requires little work if only a subset of cameras was repositioned, e.g., when someone accidentally touches a camera or when a camera is moved to a different location.

We use an algorithm called POSIT, proposed by Dementhon and Davis [21] to perform extrinsic calibration. The method can detect and match four or more non-coplanar feature points of an object in the image, of which their absolute geometry on the object is known. The method combines two algorithms. The first algorithm, POS (Pose from Orthography and Scaling), approximates the perspective projection with a scaled orthographic projection and finds the rotation matrix and the translation vector of the object by solving a linear system. The second algorithm, POSIT (POS with ITerations), uses its iteration loop, the approximated pose found by POS, to compute better scaled orthographic projections of the feature points. It applies POS to these projections instead of the original image projections. POSIT converges to accurate

pose measurements in a few iterations.

Figure 2.7 shows an example of the output from our multi-camera calibration program. In this situation, we used seven non-coplanar points. The centres of the red circles correspond with the positions that the user indicates. The little white circles represent the projections of the model point using the estimated camera parameters. In case of a good calibration, the white dots should coincide with the centre of the corresponding red circle. Reasons for inaccurate calibration are errors in intrinsic parameters (such as distortion coefficients, focal length or centre of projection), inaccurate measurements of the model points in the world coordinate system or poorly marked positions of the model points in the camera images by the user. In the example, the white dots lie inside the red circles. However, the bottom-most white dot is very close to the circle's edge. In this particular case, the reason is the rather large distance between the marked points combined with an inaccurate intrinsic calibration.

2.2 Camera configurations

Shape-from-silhouettes will be explained in more detail in Chapter 4. Two crucial criteria need to be met to apply this method: first, all silhouettes must originate from the same object, and second, we must know the view relations. There are two popular approaches to meet these criteria: capture an object from multiple views simultaneously in a calibrated camera setup (so that the object's silhouettes are projections of the same 3D object) or use a turntable to capture an object from multiple viewing directions. The latter approach is used to reconstruct static objects because the reconstruction of a moving object complicates this process significantly. In this section, we discuss both multi-camera setups and turntable setups in more detail.

2.2.1 Multi-camera setups

Most of the use cases in this work are about reconstructing moving objects or people in a scene. Therefore, we most often used multiple cameras that capture images simultaneously from different viewing angles to reconstruct the 3D shapes of the objects in the scene. Several parameters are essential when designing a multi-camera setup: the number of cameras, their position and orientation, the expected frame rate, and the image resolution. Usually, a trade-off exists between the image

(a) COSMO Mariasteen *(b) COSMO CNHi*

Figure 2.8: *Different multi-camera setups from two use cases in this work related to the COSMO project. Cameras are indicated with circles on the overview pictures.*

resolution and the frame rate. The bandwidth is often limited, even when the images are stored locally on the camera for offline processing.

The optimal position and orientation of cameras depending on the objects in the scene. Since the goal is to reconstruct moving objects, these optimal positions will be different for each different position of the objects in the scene. Therefore a camera setup is inherently suboptimal. Thus, the selection of optimal camera configurations (camera locations, orientations, etc.) for multi-camera networks remains an unsolved problem [58].

However, some rules of thumb should be considered while designing a multi-camera network for the 3D reconstruction applications in this work. Points need to be observed from different camera views to reconstruct their 3D position. Therefore, the first rule is that the field of view (FOV) of different cameras should overlap to some extent. This overlap is easily achieved when all cameras are placed next to each other and oriented to the scene. However, maximal overlap is usually not optimal. For some applications, the cameras are positioned in such a way that they observe the scene from different viewing angles. The reason is twofold: cameras in such a configuration are less likely to suffer from occlusions simultaneously, and the maximal distance between objects in the scene and cameras is limited, which causes fewer errors in computer vision techniques. Distant objects appear smaller, show less details and are harder to recognise compared to close-by objects. It is essential to have cameras at different positions in the scene to avoid a systematic error dependent on the distance between the object and the cameras to cope with this effect.

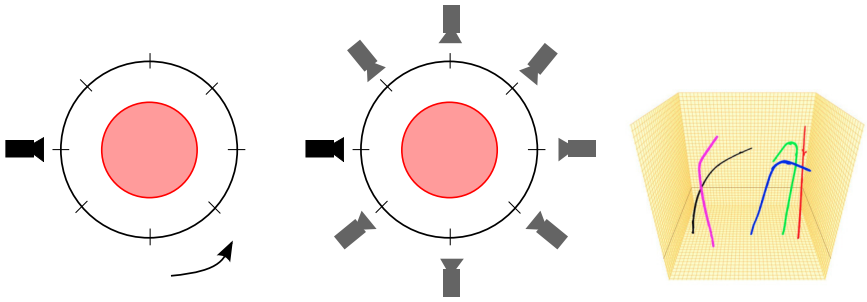
In general, cameras are mounted outside of the reconstruction volume at different heights. The reconstruction volume is defined as the part of the

scene where we will reconstruct the objects. Two camera setups from real-world use cases in the COSMO project are shown in Figure 2.8. In both cases, GoPro cameras were used to avoid wiring in these temporary setups. The reconstruction volume is significantly larger for the use case at CNHi compared to the use case at Mariasteen. Therefore, we used 5 cameras at Mariasteen and 11 cameras at CNHi. At Mariasteen all five cameras observe the complete working space of the operator, which is not the case at CNHi. The shape of the reconstruction volume and narrow spacing limited the options for mounting cameras so that the entire scene was visible. Only the three top view cameras can capture the entire scene, while the other cameras are restricted to a smaller part of the scene. Therefore this dataset is challenging in multiple aspects. The overview cameras are used for reconstructing the operator, while the other cameras are used for event detection.

2.2.2 Turntable setups using a single camera

In Figure 2.9 we illustrate how a turntable can be used to reconstruct a static object. The object is placed in the middle of the turntable. The camera captures an image of the object at different turntable positions (indicated by the dashes on the outer circle). In the example, we choose 45° as a step between each turntable position. The process generates seven additional views with a single camera. Smaller steps mean more silhouettes and, in general, a more precise reconstruction. A turntable can, therefore, generate more views than a fixed camera setup and estimate a more accurate 3D reconstruction. However, the quality depends on the correspondence between the calibrated camera settings and the exact positions of the turntable.

A turntable setup is considerably cheaper than a multi-camera setup because a single camera can generate multiple virtual views. However, the technique is only valid for static objects in specific situations: the camera must capture the object from multiple positions relative to the object, either using a turntable or moving the camera itself around the object. The use case of a turntable is different from that of a multi-camera setup, which is more flexible and more expensive. We have used a turntable in a project to reconstruct seedlings and plants. We refer the the dissertation of Simon Donné [25] for more information.



(a) Turntable setup *(b) Extra virtual views* *(c) Seedling reconstruction*

Figure 2.9: *Reconstruction of static objects using a turntable combined with a single camera. The different positions of the turntable create views from different angles that can be used together to reconstruct a static object in detail. In this case, we capture an image every* $45°$*. As an application, we reconstructed seedlings using this method for growth analysis.*

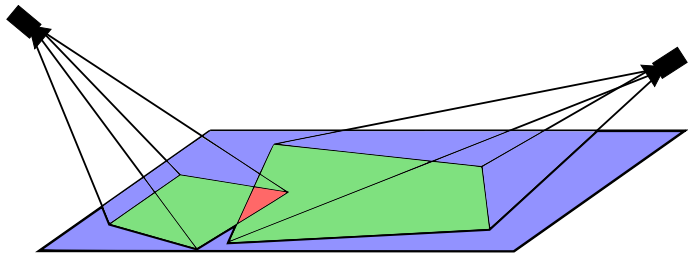


Figure 2.10: *Camera coverage visualization based on camera calibration and field of views. In this example, there is limited overlap between the two camera views, blue region: no camera coverage, green region: camera coverage by one camera, red region: camera coverage by two cameras.*

2.3 Camera coverage maps

To obtain accurate 3D reconstruction, it is important that the cameras are well-spread in the scene so that the objects in the scene can be observed from multiple different viewpoints. A camera coverage map is a tool to analyse the field of view of each camera and the camera overlap. It can be used to analyse the number of cameras that cover certain areas and hence help to decide where extra cameras should be installed to improve the camera coverage in these areas. In Figure 2.10 we show an example of a coverage map.

In Figure 2.11, the coverage map is shown for a traffic intersection dataset in Ghent. The area observed by all cameras is rather small. Also, the camera coverage maps does not take into account that objects

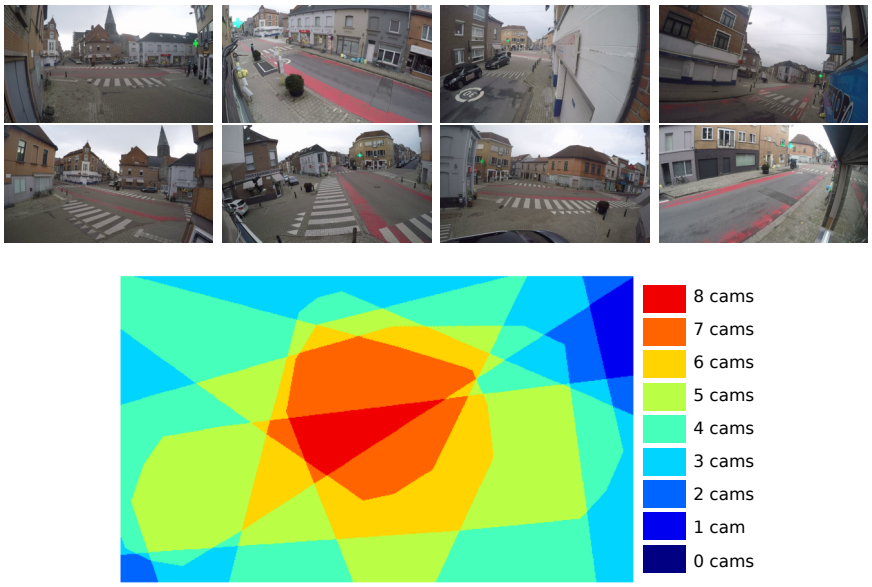


Figure 2.11: *Camera coverage map of the traffic intersection dataset in Ghent using eight cameras. The colours represent how many cameras can theoretically observe each position on the ground plane.*

such as walls from buildings limit the area that a camera can observe. Such objects create occlusion, which is one of the main focusses of this work. Knowing what a camera can and cannot see will prove to be the key to handling occlusion properly. We refer to Chapter 5 for a thorough analysis of occlusion. In that chapter, the concept of camera coverage maps is also extended to include occluders. As such, the actual camera coverage can be modelled more precisely and possible areas with limited coverage can be identified more reliably.

2.4 Sensitivity analysis of calibration accuracy

Poor camera calibration poses a challenge to 3D reconstruction. When the 3D world does not map correctly to the image, the reverse task of 3D reconstruction produces inaccurate result.

This section will illustrate the impact of inaccurate camera calibration and show how these errors manifest themselves. We have explained how a 3D point projects on an image sensor using a perspective transformation. By adapting the projection matrix in realistic ways, we can analyse the impact of such changes. To illustrate this we will project a 3D model

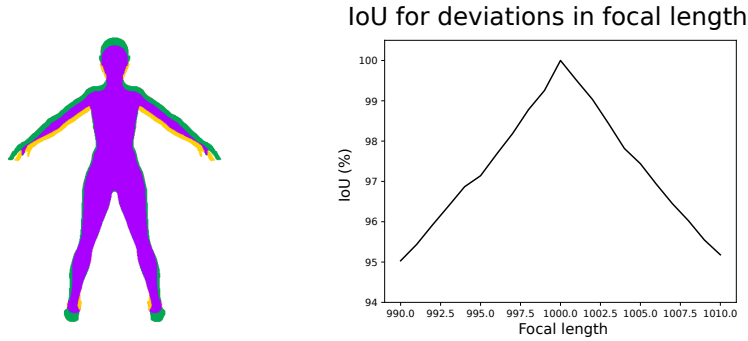
positioned 3 m in front of the camera on the camera sensor and consider the projected surface (silhouette). We compare the silhouette obtained with the initial (ground truth) and adapted projection matrix in terms of overlap. The intersection over union (IoU) forms an excellent measure to assess the impact of common errors made by the camera calibration and the effect on shape reconstruction.

Cameras in a multi-camera network can either be static or dynamic (PTZ). Static camera calibration is most convenient. The errors are usually smaller because we calibrate these cameras offline with reliable methods before they are operational, and the calibration remains fixed as long as the circumstances do not change drastically. Temperature changes may cause parameters in a camera to change. Unfortunately, cameras are sometimes moved by accident. In both cases, recalibration may be required.

PTZ cameras can change their viewing direction: pan, tilt and zoom. Pan and tilt are related to the transformation matrix (extrinsic parameters) while zooming impacts the intrinsic camera matrix. Most PTZ cameras report their motor values so that the pan, tilt and zoom positions are known, but we noticed that these values might be different from the actual positions. Also, the camera may not report the motor values fast enough to obtain reliable values for each frame, and the correct information can be delayed. In this section, we investigate the impact of errors caused by pan, tilt and zoom.

2.4.1 Zoom errors

We consider a PTZ Optics 30x NDI|HX camera that was used in one of our projects. This PTZ camera maps 16,000 motor values on 30x optical zoom. We noticed little issues with the zoom accuracy. However, while the camera is zooming, a delay exists between the reported values and the actual values. We illustrate the impact of these zooming differences. In this experiment, we simulate a camera with a focal length of 1000 px, and we compare the projected silhouette using the intersection over union for a range of 990-1110 px. In Figure 2.12 the results are shown, and we see that the IoU drops drastically both when the focal length is too large or too low. However, when the camera calibration is performed with enough checkerboard patterns (~ 100), the focal distance is usually accurate within 10 pixels [98], which corresponds to an IoU of more than 95%. Comparable results can be expected from a delay in PTZ motor values depending on the zoom speed.



(a) *IoU visualisation* ***(b)*** *IoU values for different zoom factors*

Figure 2.12: *Simulation of a focal length error using intersection over union (IoU). Purple indicates the intersection of both silhouettes, yellow and green (ground truth) are part of one of the silhouettes only.*

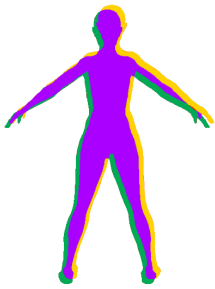
Another issue with zooming was observed during experiments. We noticed that the camera tilts down at large zoom levels. The lens moves away from the camera's centre of gravity and causes an imbalance in the camera. The motor values of the cameras do not report this offset. However, we cannot solve this type of error by adapting the zoom factor but need adjustments in the pan and tilt angles.

2.4.2 Pan/tilt errors

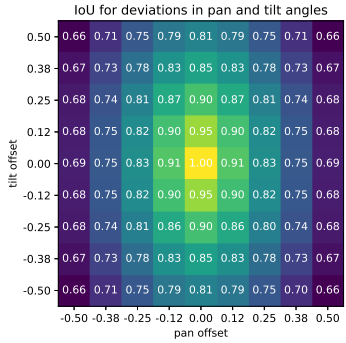
The pan and tilt angles define the orientation of the camera (assuming zero roll). This section will show how errors in pan and tilt angles manifest themselves in the camera projection. We consider errors between $-0.5°$ and $0.5°$ for both pan and tilt angles and visualise the error as a heat map in Figure 2.13. The errors in the pan direction cause relatively larger errors compared to the tilt direction. This is caused by the shape of the silhouette. Vertical edges of the silhouette's contour are more prominent. If the person were positioned horizontally, the tilt errors would be relatively larger than the pan errors. Both errors are significant, even when the error is just $0.1°$.

2.4.3 Distance-related errors

Note that the impact of zoom, pan and tilt errors also depend on the object's distance to the camera. The further away an object is positioned from the camera, the larger the error introduced by pan and tilt errors.



(a) $(\Delta pan, \Delta tilt) = (-0.5°, -0.5°)$



(b) IoU error range [-0.5,0.5] for pan and tilt

Figure 2.13: *Simulation of pan/tilt error using intersection over union (IoU). Purple indicates the intersection of both silhouettes, yellow and green (ground truth) are part of one of the silhouettes only.*

The impact of zoom errors, on the other hand, is more prominent when the object is close to the camera. The simple experiments in Section 2.4.1 and 2.4.2 show that camera calibration is essential to achieve a reliable result. They also show how we can model these errors given a particular multi-camera setup.

2.5 Conclusion

This chapter explained the different camera parameters and how to estimate them using camera images of calibration objects. The intrinsic calibration is performed using a planar pattern, such as a checkerboard in front of the cameras. The extrinsic camera parameters are estimated based on the projections of a known calibration object using the POSIT algorithm. A fully calibrated camera setup can aid to create a camera coverage map to analyse how many cameras cover each area. Camera calibration is an important step towards reliable 3D reconstruction. The experiments show that camera calibration is sensitive to zoom, pan and tilt errors and depends on the particular camera configuration.

The main contributions made in this chapter are the following:

- the use of the POSIT algorithm as a camera calibration tool for extrinsic camera parameters;
- the creation of camera coverage maps based on a calibrated multi-camera setup;

- the development of a software program to calibrate cameras intrinsically and extrinsically for a multi-camera setup, both online and offline.

Some of the work in this chapter is described in some publications:

- J. Guan, F. Deboeverie, M. Slembrouck, D. Van Haerenborgh, D. Van Cauwelaert, P. Veelaert, and W. Philips. Extrinsic calibration of camera networks based on pedestrians. *Sensors*, 16(5), 2016. ISSN 1424-8220
- J. Guan, F. Deboeverie, M. Slembrouck, D. Van Haerenborgh, D. Van Cauwelaert, P. Veelaert, and W. Philips. Extrinsic calibration of camera networks using a sphere. *Sensors*, 15(8):18985–19005, 2015. ISSN 1424-8220
- D. Van Hamme, M. Slembrouck, D. Van Haerenborgh, D. Van Cauwelaert, P. Veelaert, and W. Philips. Parameter-unaware autocalibration for occupancy mapping. In *Proceedings of the 7th international Conference on Distributed Smart Cameras (ICDSC'13)*, pages 49–54. IEEE, 2013. ISBN 9781479921669

Chapter 3

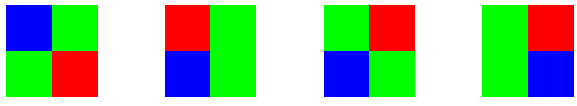
Silhouette Extraction

We try to understand the scene in front of the multi-camera setup by reconstructing objects in the scene. To do so, we need to localize these objects in the different camera images. We represent these objects in the images as silhouettes of the objects in the scenes. Together with the multi-camera calibration, we use these silhouettes to reconstruct the objects in 3D. The most qualitative volumetric reconstruction is obtained when both the calibration and the silhouettes are accurate and precise. Silhouettes can be either too small, e.g., part of an object that is not detected, or too large, e.g., objects in the vicinity, are mistakenly considered to belong to another object.

Silhouettes are obtained using image segmentation, which is the process of partitioning a digital image into multiple segments (sets of pixels). The goal of segmentation is to simplify and change the representation of an image into something more meaningful and easier to analyse. More precisely, image segmentation is the process of assigning a label to every pixel in an image such that pixels with the same label share specific characteristics. In our case, each pixel is either labelled as part of the silhouette (foreground label) or not (background label), which can be represented as a binary image.

We focus on three different image segmentation methods in this chapter, each having its own application field: colour segmentation, motion segmentation and neural network segmentation. In Section 3.4, we discuss which method is best in which circumstances.

In Chapter 5 the problem of incomplete silhouettes will be discussed in more detail, but for now, we focus on extracting the visible part of objects as accurately as possible.



(a) *BGGR* *(b)* *RGBG* *(c)* *GRGB* *(d)* *RGGB*

Figure 3.1: *Bayer patterns: BGGR, RGBG, GRBG and GRBG*

3.1 Colour segmentation

Before we can segment colour pixels from images, it is essential to know how colour is represented in digital images. Such images are composed of three channels: one for the red light, one for the green light and one for the blue light. Each pixel (u, v), therefore, has three values expressing the amount of red, green and blue light needed to reproduce its colour. A broad spectrum of colours can be reproduced in this way. We will not go into further detail as it is not relevant for this work.

Cameras capture these colour values by directing the incoming light through a red (R), green (G) or blue (B) filter and measuring the amount of light that hits the sensor at each location for each pixel. In reality, the camera sensor usually does not measure the amount of light for each channel per pixel but uses a so-called Bayer pattern where R, G and B filters are alternated in specific patterns (see Figure 3.1), presented in [7]. The bayer patterns are 50% green, 25% red and 25% blue. The green photosensors are the luminance-sensitive elements, and the red and blue ones are the chrominance-sensitive elements. There are twice as many green elements as red or blue to mimic the physiology of the human eye. Only red, green, or blue light is measured at each pixel location on the image sensor. A debayering process interpolates the missing values so that for each pixel value, an R, G and B value is obtained.

A typical use case where colour segmentation can be used is when either the background or the object itself has a distinctive colour. For instance, a weatherman is recorded in front of a green screen (green key recording) so that the green colour can be removed and he can be overlayed on the weather map, as shown in Figure 3.2.

However, filtering a colour is not as straightforward as one may think. Colour is rarely constant over the entire image, and there are lighter and darker shades that should be included. For instance, as illustrated in Figure 3.2 and 3.3, in front of a green screen there may be multiple shades of green in the background. A number of these pixels are extracted and enlarged in Figure 3.3. The difference is rather substantial. The green

Figure 3.2: *A typical use case where colour segmentation can be used to extract the weatherman.*

RGB	36,116,57	18,79,38	88,77,63	166,50,46
HSV	96,176,116	99,197,79	24,72,88	1,184,96
RGB	21,125,54	26,108,46	28,37,40	127,105,93
HSV	98,212,125	95,194,108	138,77,40	15,68,127

Figure 3.3: *Different RGB values and corresponding HSV values in the background and foreground of Figure 3.2. Four pixels were manually picked from different parts of the background. The other four pixels are part of the hair, skin, suit and tie of the weatherman.*

component is larger than the red and blue values, but some of the pixels in the foreground also have strong responses in the green channel.

Therefore, other colour spaces are often considered for colour segmentation. The HSV colour space represents colours in three new channels: hue, saturation and value. HSV separates the luminance (saturation and value) from the chrominance (hue). One of the advantages of such a colour space is robustness against light changes, mainly affecting the intensity channel. In Figure 3.3 all green values can be effectively separated from the foreground value by using the 95-99 range in the hue channel. Note that we have mapped both RGB and HSV values to 8 bit per channel for simplicity.

Colour segmentation works best when the colour ranges of the foreground and background do not overlap. Unfortunately, the technique is often not suitable in real-world environments since the background rarely consists of a colour range that does not exist in the foreground. Since we are interested in moving objects/persons, motion segmentation techniques can be used. These methods do not require a uniform and

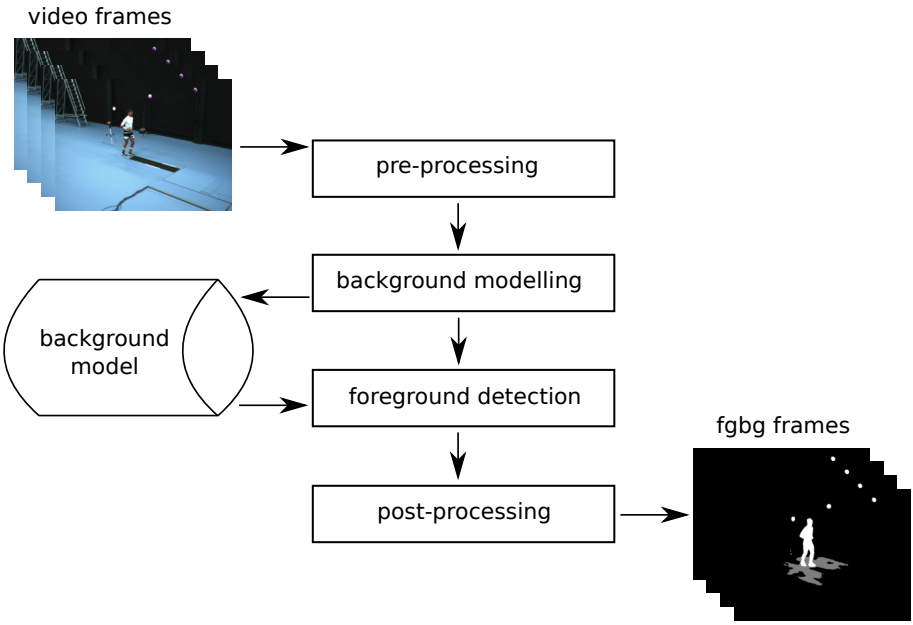


Figure 3.4: *Foreground/background segmentation overview. Typical foreground/background segmentation methods use a background model which is compared to new images and adapted accordingly. Pre-processing and post-processing operations reduce noise in the generated masks.*

constant background because these methods model the background for each pixel individually and update these models when light conditions are changing.

3.2 Motion segmentation

Motion segmentation is often referred to as foreground/background segmentation. Its purpose is to separate a moving object (foreground) from the more or less static objects (background) in a scene. Motion segmentation methods model the appearance of the background through learning based on examples. After the learning phase, an algorithm classifies pixels as foreground or background by comparing them with the model. We visualise the data flow in figure 3.4. The pre- and post-processing steps represent simple operations on the images to improve detection, such as smoothing and morphological filters to reduce noise. Two image features are commonly used to model a background: colour and edges.

Background models based on colour perform well in indoor environments as long as the appearance of the foreground object is dissimilar enough from the background objects. Typical issues arise with dark hair in front of a dark-coloured wall or light coloured clothing in front of a light-coloured wall. Such methods typically perform less accurate when the light in the scene changes. While light can be controlled to a certain extent indoors, outdoor environments pose extra challenges, especially in cloudy weather or capturing data over a long time.

Edges are more stable than colours during light changes and are often used to track movement in outdoor scenes. These methods learn the location of static edges in the scene. After the learning process, edges that do not correspond to the learned static edges are treated as foreground objects. However, since only the edges are tracked, a filling algorithm is needed to created actual silhouettes.

Multiple criteria are important for our application: processing speed, precision and recall. Processing speed is vital because frames should be processed in real-time (minimum 25 fps). Precision is essential so that the pixels classified as part of a silhouette are part of the object's projection. Even more critical is the recall. When a method fails to detect parts of the moving object, the silhouettes become incomplete, which will cause missing parts in the 3D shape reconstruction if handled poorly.

Occluders also cause incomplete silhouettes in part of the scene, but their nature is different from the incompleteness that comes from erroneous foreground/background segmentation. When an occlusion occurs, the errors may not be present for all objects in the same image location and, therefore, may not have a common characteristic that can be modelled statistically. In Chapter 5 we go into more detail on how to solve reconstructions from incomplete silhouettes, but first, we focus on detecting the parts of the moving objects that are visible by the camera as well as possible.

We have used multiple motion segmentation algorithms in this work. Table 3.1 summarizes the ones that were used most frequently. To understand the difference between these methods, we give more context in the following paragraphs.

The C-EFIC method proposed by Allebosch et al. [1] uses both colour information and edges to model the background. The method performs well and outperforms most methods in challenging circumstances such as low-light videos and varying illumination in general, which typically occurs in outdoor scenes. However, the method is less suitable for real-

Name	Ref	Description
C-EFIC	[1]	Colour and edge-based adaptive background model with interior classification
ViBe	[3]	Sample-based adaptive background model with random characteristics
Kim	[49]	Static background model using fixed colour threshold
SuBSENSE	[92]	Local sensitivity using a adaptive background model
MoG Zivkovic	[114]	Adaptive mixture of Gaussians as background model
Petrovic	[33]	Short-term and long-term adaptive background model

Table 3.1: *Different foreground/background segmentation methods used in this work. These methods represent different approaches: static background model (Kim), single dynamic background model (C-EFIC, ViBE, SuBSENSE, MoG Zivkovic) and a combination of a short-term and long-term background model (Petrovic).*

time processing. They report a processing speed of ±16 fps for a 320x240 video with C++ code running on an E5 Quad Core. We typically use 780x580 resolution with a minimum of 4 camera streams that need to be processed. Therefore, this method is only used in applications that are not time-critical. We used this method primarily to analyse a traffic intersection based on a 2-hour video recording with 11 cameras.

ViBe is a foreground/background segmentation method that stores the background model in RGB value samples [3]. ViBe does not have a dedicated learning period but randomly replaces old samples with new samples in the pixel's neighbourhood to keep the values of the background up-to-date. The method can handle small camera movements using the neighbourhood of a pixel instead of the pixel's location. Pixel values that do not correspond to any of the model's samples at the corresponding pixel location are classified as foreground. The method is suitable for indoor and outdoor scenes and has been patented. For our application, however, ViBe seems to be less suitable. Often, parts of the silhouette are missing, and the level of noise is bothersome in 3D reconstruction.

The method of Kim et al. [49] is not the most complicated nor the most suitable for a wide range of applications. However, it performs well in indoor scenes with constant illumination, where recordings are typically

just a few seconds long and when it is possible to record an empty scene to build a reliable background model. It is a standard method that learns the background from several frames without motion, and the background is not updated afterwards. The mean and standard deviation of the luminance and the colour (hue) are determined for each pixel. Multiple thresholds on the luminance decide whether a pixel belongs to one of four classes: (a) reliable background, (b) suspicious background, (c) suspicious foreground of (d) reliable foreground. The suspicious foreground pixels are also compared with the mean and standard deviation of the hue to detect shadows and change the class from (c) to (b) instead. The method works especially well with low-noise cameras such as the Allied Vision Technology ethernet cameras we often use. The expected frame rate is 30 fps for a 1024×768 RGB stream on a PC with a Pentium IV 3.2 GHz CPU, 1 GB memory.

SuBSENSE [92] once was at the top of the best performing motion segmentation benchmarks, such as CD.net 2014 [32]. SuBSENSE is a universal pixel-level segmentation method that relies on spatio-temporal binary features and colour information to detect changes. Such a model allows camouflaged foreground objects to be detected more easily while most illumination variations are ignored. SuBSENSE uses pixel-level feedback loops to adjust its internal parameters dynamically. The expected frame rate on a third-generation Intel i5 CPU at 3.3 GHz is between 45 and 90 fps, depending on whether the feedback loop is used.

One of the older FG/BG methods is one based on Gaussian Mixture Models (GMM). This method models the different background colours as a mixture of Gaussian distributions. The proposed method by Zivkovic [114] is particularly useful because it is both adaptive and detects shadows. Shadows are often mistakenly detected as part of the foreground because the lack of light significantly changes the colour of these pixels. Thanks to a shadow detector, these pixels can be removed from the silhouettes. This method is helpful in indoor scenes with slight changes in the background. It can also handle displacement of a static object as those objects are learned into the background after some time. When a person stands still for too long, he becomes parts of the background model and is no longer detected as foreground. The processing time is around 49 fps for 720×480 images on a PC with Core i7 3.4 GHz, which makes it suitable for real-time applications.

The method of Grünwedel and Petrovic [33] is computationally cheap and has low memory requirements. They combined an autoregressive moving average filter with two background models having different adaptation speeds. The first model, having a lower adaptation speed, models

long-term background and detects foreground objects by finding areas in the current frame, which significantly differ from the proposed background model. With a higher adaptation speed, the second model represents the short-term background and is responsible for finding regions in the scene with a high foreground object activity. The final foreground detection is built by combining the outputs from these building blocks. The foreground obtained by the long-term modelling block is verified by the output of the short-term modelling block, i.e. only the objects exhibiting significant motion are detected as natural foreground objects.

Some applications require tailor-made solutions. For the IPLAY project, we investigated whether it was possible to segment the silhouette of a person standing in front of a projection screen [2]. The contents of the projection screen were known. Using colour-tone mapping from the colours sent to the screen and the appearance of those colours on the cameras that captured it, we could model the background. The method proved effective, but only in controlled circumstances. A change in ambient light, for instance, required a recalibration of the colour-tone mapping. The process ran too slow for real-time processing (around 10 fps), although it was not optimised to increase computation speed. Even under these strictly controlled circumstances, foreground/background segmentation remains challenging. Later in the project, we discarded foreground/background segmentation and focussed on pose extractors instead because this approach handles the player's pose directly and pose extractors are less sensitive to variations in ambient light or changes in background pixels. Pose extractors are slower than foreground/background segmentation methods, but by parallelising the computations on multiple GPUs, the total computation speed remains real-time (see Chapter 6).

3.3 Neural network segmentation

Although motion segmentation methods produce reliable silhouettes, they are usually not applicable when the object is standing still for a long time or when the background is changing too much, which might be the case in outdoor scenes due to illumination changes.

Neural network segmentation methods are advanced object detectors that may also segment detected objects into meaningful parts. This segmentation corresponds to the silhouettes that we are investigating in this chapter. The processing speed of such methods depends mainly on the available GPU and the complexity of the neural network. However,

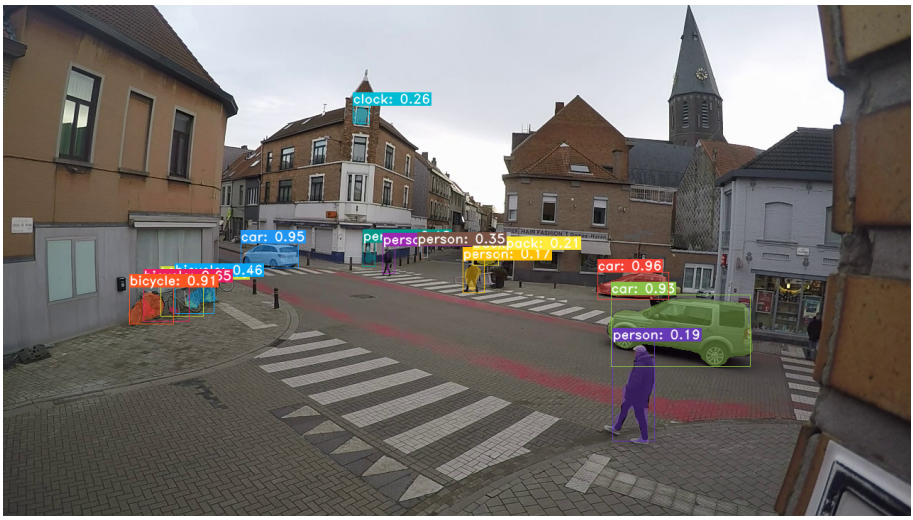


Figure 3.5: *YOLACT neural network segmentation on an image from the traffic intersection dataset in Ghent. Multiple objects are detected and segmented such as a vehicles, bicycles and persons. The person with the purple label near the centre of the image is detected but not segmented. The quality of the segmentation is correlated to the size of the object in the image.*

they require more resources than the traditional methods discussed in the previous sections. Therefore, it might not always be feasible to use them in a real-time setting without a powerful GPU. Also, these methods only work for object classes that were part of the training set, which may be a problem if we are interested in fewer everyday objects. An example of neural network segmentation can be seen in Figure 3.5.

Object segmentation is challenging because it needs to detect an object correctly and also segment it precisely. Such methods combine object detection in the form of bounding boxes and semantic segmentation to classify each pixel within a detected bounding box. Object detection networks such as Faster R-CNN [77] use a region proposal network (RPN) in the first phase to detect meaningful parts in the image, after which each of these regions is further investigated.

Mask R-CNN [41] extends Faster R-CNN by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. The mask branch is a small, fully connected network (FCN) applied to each ROI, predicting a segmentation mask in a pixel-to-pixel manner.

Mask R-CNN is still seen as a baseline method for further research. More recent advances were able to improve both the mask segmentation

further and lower the computation time. YOLACT [10] for instance, processes frames at least 2.5 times faster than mask R-CNN but with decreased precision. Such results allow real-time usage, which is essential for our application.

YOLACT breaks instance segmentation into two parallel subtasks: (1) generating a set of prototype masks and (2) predicting per-instance mask coefficients. Then they produce instance masks by linearly combining the prototypes with the mask coefficients. Because this process does not depend on repooling, this approach produces very high-quality masks and exhibits temporal stability for free.

Even more recent is BlendMask [17], which claims improved mask prediction by effectively combining instance-level information with semantic information with lower level fine-granularity. Their main contribution is a blender module that draws inspiration from both top-down and bottom-up instance segmentation approaches. Their light-weight version achieves 25 fps on a single 1080Ti GPU.

We compare Mask R-CNN, YOLACT and BlendMask against more conventional silhouette extractors in the next section.

3.4 Experiments and discussion

This section discusses how we decide which silhouette extractor to use. There are two essential factors for this decision: the processing speed and the quality of the produced silhouettes. Processing speeds are only necessary when real-time processing is demanded. For offline processing, computation speed is less of an issue. The quality of the produced silhouettes, on the other hand, is always critical.

For the 3D reconstruction application, the detected silhouettes must contain the entire visible area of the objects. When the silhouettes are consistently too oversized in all camera views, the 3D reconstruction will also be too large and lack finer details on the object's surface. Silhouettes that are too small or contain holes result in 3D shapes where part of the object is not reconstructed. Therefore, the silhouettes must be detected as precisely as possible.

An experiment was conducted with several silhouette extractors to assess both the computation speed and the quality of the silhouettes. The experiment was performed on two different datasets: one indoor dataset of a runner in a lab and one outdoor dataset with cars, cyclists and pedestrians on an intersection.

Gound truth silhouettes were manually produced by segmenting the moving object in the frames and converting these frames to binary images for both datasets. The different methods were executed on the same computer with an Intel Core i7 CPU 960 @ 3.2 GHz x 4 with 23.5 GB of RAM and an RTX 2070 GPU to make a fair comparison between the different methods. The reported speeds are for processing one video at a time. The CPU-based methods use a single core so that processing multiple frames in parallel does not impact the speed too much for less than four parallel streams.

Let $v_g(x, y)$ represent the value of the silhouette on position (x, y) of the binary image. If 1, pixel (x, y) belongs to the silhouette, if 0, it does not. Let $v_m(x, y)$ represent the value of a silhouette product by method m. There are four important measures that can be calculated per pixel between a ground truth silhouette and a silhouette produces by one of the methods:

- TN: true negative if $v_g(x, y) = 0$ and $v_m(x, y) = 0$;
- FN: false negative if $v_g(x, y) = 1$ and $v_m(x, y) = 0$;
- TP: true positive if $v_g(x, y) = 1$ and $v_m(x, y) = 1$;
- FP: false positive if $v_g(x, y) = 0$ and $v_m(x, y) = 1$.

These measures are evaluated for each pixel in the ground truth. By taking the sum over all pixels, we calculate the recall (Re) and the precision (Pr). Note that high recall values indicate that the silhouettes contain limited holes and high precision values indicate that of there are limited excess pixels detected as being part of a silhouette compared to the ground truth:

$$\text{Re} = \frac{\text{TP}}{\text{TP} + \text{FN}}, \tag{3.1}$$

$$\text{Pr} = \frac{\text{TP}}{\text{TP} + \text{FP}}. \tag{3.2}$$

Both recall and precision are important. The F1-score expresses the trade-off between the two in a single number:

$$\text{F1-score} = \frac{2\ \text{Re}\ \text{Pr}}{\text{Re} + \text{Pr}} \tag{3.3}$$

However, in Chapter 5, precision becomes more critical because the incomplete parts (low recall) are compensated for by the occlusion handling process.

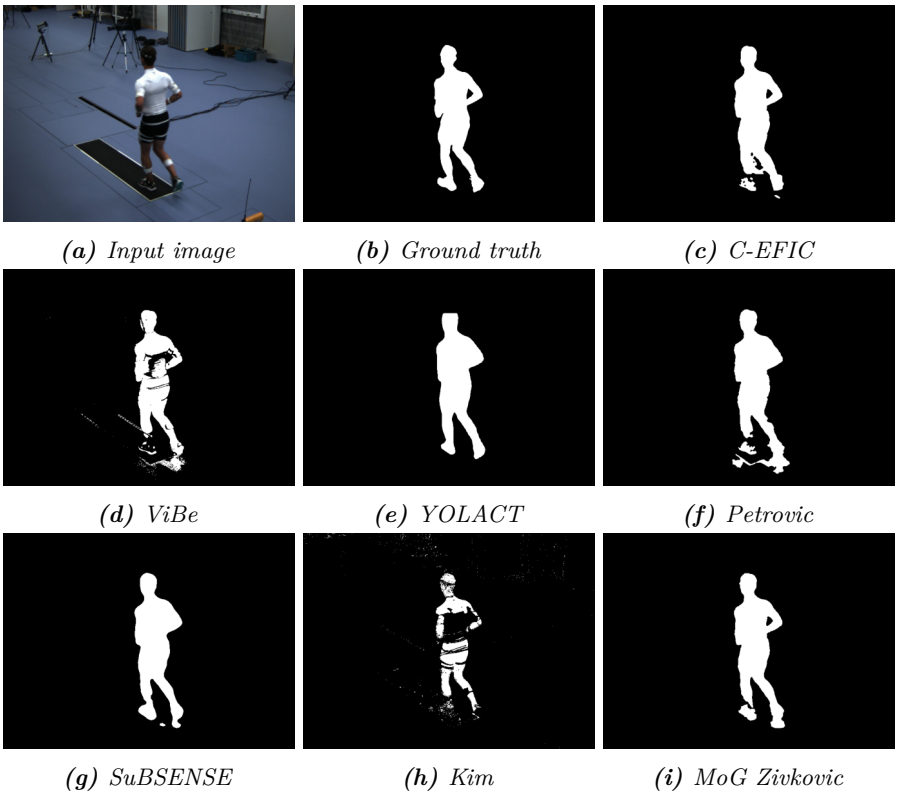
Method	FPS	Re	Pr	F1
C-EFIC [1]	2	.90	.96	.93
ViBe [3]	107	.93	.81	.86
YOLACT [10]	33	.90	.93	.91
Petrovic [33]	53	.87	.96	.91
SuBSENSE [92]	3	.92	.95	.93
Kim [49]	144	.80	.88	.84
MoG Zivkovic [114]	40	.96	.93	.94

Table 3.2: *FPS and F1-scores for segmentation results on the SSL-JR dataset. MoG Zivkovic achieves the highest F1-score while reaching 40 fps. C-EFIC and SuBSENSE perform almost as good, but only run at 2-3 fps which is too slow for real-time applications.*

3.4.1 Experiment 1: indoor environment

The first experiment is performed on an indoor dataset where a person runs from one side to the other side in the Sports Science Lab Jacques Rogge using eight ethernet cameras (type AVT Manta) at 67 fps with a resolution of each 780x580 pixels. In Table 3.2 and Figure 3.6, we see the results using this dataset. Apart from C-EFIC and SuBSENSE, all methods can run in real-time. However, there are significant differences in precision and recall between the methods. MoG Zivkovic performs best in terms of recall and F1-score. Petrovic and C-EFIC perform best in terms of precision, but the low recall will cause missing parts in the 3D reconstruction. MoG Zivkovic is an overall good choice for silhouette extraction in an indoor environment, and with 40 fps on a single processor core, it is possible to keep it real-time, even if there are four or more camera feeds to be processed.

Note that the tables show results for a single camera view. Our 3D reconstruction technique aims to reduce errors in the provided silhouettes by combining multiple views. Whether or not that is possible will also depend on the correlation between the errors from different views. If the same part of an object is not segmented in the silhouettes of multiple views, the part will be reconstructed with fewer details or not at all. Therefore a camera network should be designed such that errors are as uncorrelated as possible, for instance by making sure that the background colour is not the same in all views. In this particular case, the shadow's colour on the runner's shirt is similar to the blue colour of the floor. Some methods, such as Kim cannot segment the person correctly. Therefore, it is advised to make sure that background is not



(a) Input image *(b) Ground truth* *(c) C-EFIC*

(d) ViBe *(e) YOLACT* *(f) Petrovic*

(g) SuBSENSE *(h) Kim* *(i) MoG Zivkovic*

Figure 3.6: *Visual segmentation results of the indoor experiment on the SSL-JR dataset using the different methods. These images confirm that MoG Zovkovic, C-EFIC and SuBSENSE segment the person most reliably. The shadows on the runner's shirt cause most issues because the colour is similar to the ground floor.*

the same in all camera views, unless the colour range does not overlap with the colour range of the test subject, e.g., a green key studio.

3.4.2 Experiment 2: outdoor environment

The second experiment was conducted on a dataset of a traffic intersection in Ghent. The video footage was recorded over a timespan of 2 hours with 11 GoPro cameras. The main challenges are varying light conditions and objects that are not moving for a long time, such as parked cars or pedestrian waiting for a bus. The resolution of each of the video streams is 1280x720 pixels. Therefore, the processing times are slower than those for the indoor experiment. There are also more objects in the scene. Table 3.3 and Figure 3.7 show the results for the quality

Method	FPS	Re	Pr	F1
C-EFIC [1]	1	.93	.84	.88
ViBe [3]	43	.54	.54	.54
YOLACT [10]	7	.95	.70	.81
Petrovic [33]	23	.19	.52	.27
SuBSENSE [92]	2	.68	.74	.71
Kim [49]	57	.09	.96	.17
MoG Zivkovic [114]	14	.26	.79	.39

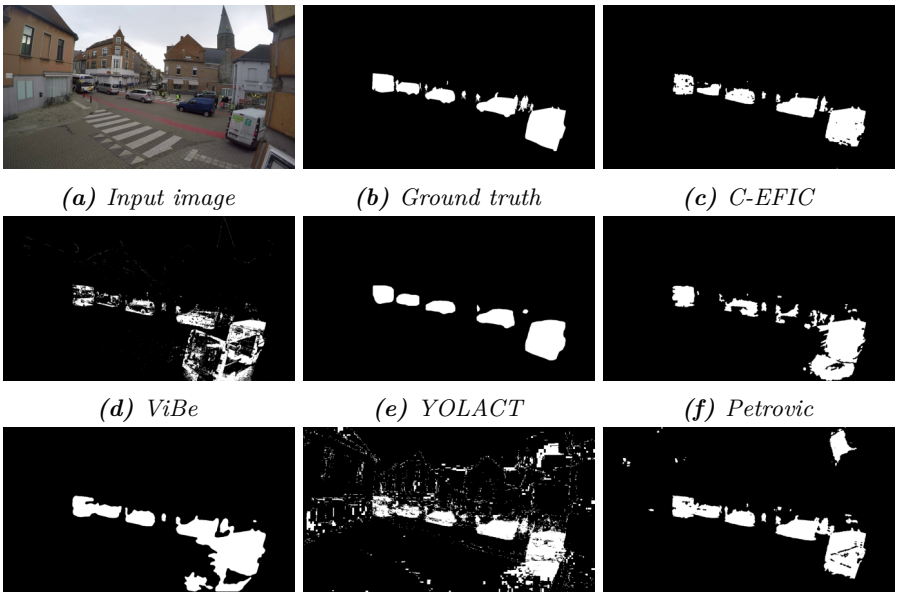
Table 3.3: *FPS and F1-scores for results on the traffic intersection dataset. The results are significantly worse compared to the indoor experiment due to typical lighting changes. C-EFIC performs the best on this dataset because of the edge-based approach. Edges are less sensitive to lighting changes. The best performing real-time method is ViBe.*

of the produced silhouettes. We used a video sequence of 10.5 minutes long and manually segmented ground truth every 30 seconds to measure the quality. There are significant illumination changes during the video due to the sun appearing and disappearing behind clouds. There are pedestrians, cyclists and vehicles moving around the entire sequence.

The best-performing methods are C-EFIC and YOLACT. Unfortunately, the implementation of both methods is currently too slow to be used in real-time because they run only at 1 and 7 fps respectively, and we need to run these algorithms on multiple images from different views simultaneously to create a real-time reconstruction. An important note to make about YOLACT is that this method also detects the separate objects while the other methods produce combined silhouette images only. Distinguishing objects from each other has many benefits. The single-object silhouettes can produce more accurate 3D reconstructions of an object as long as the cross-view corresponding silhouettes can be determined.

YOLACT does not distinguish between moving object and static objects. Therefore all objects can be detected, as opposed to motion-based segmentation methods. If we are only interested in moving object, these static objects can be removed using object trackers. If the position of objects is more important than the exact shape of the objects, YOLACT may be the best choice to map all different objects in the scene frame by frame, even if some of these objects have been stationary for some time.

Unfortunately, YOLACT is not reliable to detect all road users. The method is trained to detect persons, bicycles, cars, trucks and buses. Other objects will not be detected unless they are added to the training



(a) Input image *(b) Ground truth* *(c) C-EFIC*

(d) ViBe *(e) YOLACT* *(f) Petrovic*

(g) SuBSENSE *(h) Kim* *(i) MoG Zivkovic*

Figure 3.7: *Visual results of the outdoor experiment using the different methods. The errors are significantly larger than for indoor scenes. Small objects are often not detected and large object are often not completely segmented. Most methods create holes in the objects (ViBe, Petrovic) are suffer from severe noise (Kim). C-EFIC's segmentation is clearly the best, showing only little mistakes.*

data. It is important to take these missing objects into account, for instance, in the case of obstacle avoidance. When YOLACT does not recognise an obstacle, such as a dog crossing a street, it can create dangerous situations.

The faster algorithms cannot segment the objects in a reliable way. Therefore, we are currently restricted to analyse traffic offline using one of the slower methods such as C-EFIC, SuBSENSE or YOLACT. From this analysis, we may conclude that 3D reconstruction of objects in an outdoor environment is not trivial. Not only the silhouette segmentation, but also the camera calibration is not accurate enough to create a reliable reconstruction. However, traffic remains an interesting use case because of many occlusions that naturally occur in such environment. If computers become even more powerful and segmentation methods improve further, our techniques may also be applied in these situations. In Chapter 5, we show a simulation of what can be achieved under these circumstances.

3.5 Conclusion

This chapter discussed several different approaches to detect silhouettes in images based on motion, colour, or object classification. There is a wide range of methods, each tailored to specific use cases. Therefore, it is important to decide which method to use in each particular situation.

The indoor environment is often easier due to fewer illumination changes and more control over what is happening in the scene than the outdoor environment. We conducted an experiment that shows significant differences between both cases.

For the 3D reconstruction, it is crucial to segment the silhouettes of an object both entirely and precisely. Therefore both recall and precision are essential parameters. Both should be as high as possible. Therefore, our decision of which method to use is usually based on the F1-score, which is the harmonic mean of recall and precision.

In the experiments, we noticed that all methods performed rather well in an indoor environment, with MoG Zivkovic producing the best-segmented silhouettes followed closely by C-EFIC. In the outdoor environment, however, some methods proved to be useless while others could still detect the moving object decently. We noticed that C-EFIC produces the most qualitative silhouettes with YOLACT (neural network segmentation) as second best for outdoor environments. The complex scenes with significant illumination changes due to the sun disappearing behind clouds make it hard for less sophisticated algorithms to produce reliable results. While motion segmentation algorithms can detect objects of arbitrary shape, YOLACT can only detect objects that belong to classes that are present in the training data. When using YOLACT, it is crucial to consider this because it could lead to dangerous situations.

The results in the outdoor dataset show that object segmentation is not reliable in real-time segmentation methods. Combined with larger errors in the camera calibration, it seems that 3D reconstruction in these circumstances still poses a big challenge, and a real-time traffic reconstruction based on silhouette information with limited computation resources is not yet possible.

The main contributions made in this chapter are the following:

- an overview of three different ways to segment objects in an images based on colour segmentation, motion segmentation and neural network segmentation;

- testing and comparing different silhouette extraction techniques in different circumstances;
- an application to execute and compare different silhouette extraction methods to process videos.

Some of the work related to this chapter is described in these publications:

- G. Allebosch, M. Slembrouck, S. Roegiers, H. Luong, P. Veelaert, and W. Philips. Foreground background segmentation in front of changing footage on a video screen. In *Proceedings of Advanced concepts for intelligent vision systems (ACIVS'18)*, volume 11182, pages 175–187. Springer International Publishing, 2018. ISBN 9783030014483
- K. Bauters, H. Van Landeghem, D. Van Haerenborgh, M. Slembrouck, D. Van Cauwelaert, P. Veelaert, and W. Philips. Multi-camera complexity assessment system for assembly line work stations. In *Proceedings of the European Concurrent Engineering Conference (ECEC'13)*, page 5, 2013

Chapter 4

Volumetric Shape Reconstruction

When it comes to finding out the spatio-temporal relationship between an object and its surroundings, it is usually required to estimate its 3D shape in the 3D world. This information can be directly used in AR/VR applications or for motion analysis. For instance, when monitoring a worker at a conveyor belt over time and reconstructing his/her 3D shape, it is possible to determine analytics about the distance travelled, the number of times the worker needs to bend his/her knees or reach out for some object on a shelf. The worker's task can then be optimised for ergonomics and assembly time reduction.

Volumetric shape reconstruction can be performed with multiple techniques. Most popular approaches involve stereo pairs, depth sensors (time-of-flight cameras or LiDARs) or wide-baseline cameras. While stereo pairs and depth sensors can create accurate shape reconstructions, they do not offer full 3D reconstructions because the object is only observed from a single direction. We refer to 2.5D instead of 3D in these situations. The output can be represented as a depth map containing the distance to the closest object for each position on the depth sensor.

Multiple (single-view) depth sensors can mitigate the lack of 3D reconstruction, but such extension is not always possible or straightforward due to interference between sensors. In this work, we use RGB cameras, which are relatively cheap compared to other sensors. A stereo pair consists of two RGB cameras mounted next to each other and looking in the same direction. The disparity created by the slightly different views can be exploited to create a 2.5D reconstruction. However, since the viewing direction is almost identical, occlusion in one camera will also

be present in the other camera view of the stereo pair. It is better to view the scene from multiple directions to solve the occlusion problem. For the same number of cameras, wide-baseline camera setups offer double the number of views than stereo pairs. Alternative approaches will be discussed in the next section to support our choice for wide-baseline camera setups.

It is essential to keep in mind that the goal of the desired 3D reconstruction is twofold: determine the volumes that are occupied by the objects in the scenes, and perform further analysis on each of these objects, e.g., interaction between objects, tracking of objects, and shape analysis. Applications will be discussed in more detail in Chapter 7.

In this chapter we will explain how we combine camera calibration (Chapter 2) and silhouettes (Chapter 3) to reconstruct a 3D shape. First, we overview multiple shape reconstruction techniques based on different sensors. The second part of this chapter discusses 3D representations. The third part gives an in-depth analysis of a particular technique called shape-from-silhouettes [52]. In the last part, we discuss how such reconstruction can be further refined by using the silhouettes and the colour values of the pixels inside this silhouette (space carving [50, 18]).

4.1 Shape reconstruction from images

3D shape recovery from images is ill-posed, i.e., there may be multiple shapes that are consistent with the same set of images. Therefore, reconstruction methods must either choose one particular shape from the space of all consistent shapes or reconstruct the all encompassing volume by which all camera-consistent shapes are bound. For the first approach, a smooth surface is favoured over an irregular surface because smooth surfaces are often more realistic. However, a drawback of this type of approach is that it typically penalises discontinuities and sharp edges, which are very common in natural scenes. The second approach poses no constraints on a reconstructed shape and can handle a broader range of different shapes. However, these shapes are often larger than shapes reconstructed from the first approach, in particular when only a few cameras are used.

In literature, multiple techniques have been proposed to reconstruct shapes from input images. This section provides an overview, and we discuss each method regarding suitability for the applications. Note that we mainly consider moving arbitrarily-shaped objects and are interested in their surface geometry. Typical objects in this work are cars and

humans. For a more in-depth analysis on how to reconstruct a person's skeleton in 3D, we refer to Chapter 6.

Methods for shape reconstruction can be classified in different ways. One such division is the number of viewpoints that are used. Single viewpoint solutions are most applicable but usually less reliable, and the problem is even more ill-posed than if there are multiple views available.

4.1.1 Single-sensor reconstruction

Reconstruction methods based on data from a single sensor can only produce 2.5D reconstructions (depth-sensing) because the scene is viewed from a single viewpoint. A full 3D reconstruction is impossible without prior knowledge of the object. One of many possible reconstruction methods from a single view is the shape from shading [76]. This method observes shade variations in the image of the object and estimates the object's surface from these clues. Such a method only works in controlled circumstances with a single light source and objects coated in the same material. Alternatively, structured light can be used. The technique uses a projector (light source) which casts a specific pattern on the object, and a camera observes the deformation of this pattern by the object. Since the projected pattern and the observed pattern are known, it is possible to reconstruct the shape on which the pattern is projected. The first version of the Microsoft Kinect used structured light to generate a 2.5D representation of the objects in front of the sensor.

We also consider a stereo pair as a single viewpoint sensor because both cameras more or less observe the scene from the same viewing direction despite a particular parallax between the two sensors. Stereo cameras can be used for 2.5D reconstruction and perform well in general when the object to reconstruct has rich features which can be detected and matched between the two camera views [53]. However, real-time processing is not always guaranteed due to the computationally expensive matching algorithms which transform stereo images into depth maps.

Time-of-flight (ToF) techniques are also valuable for shape reconstruction. Both ToF cameras and LiDARs use light pulses and measure the time for the pulse to travel to the object and return to the sensor. These depth measurements can be used to create a depth map. The functional difference between LiDAR and other forms of ToF is that LiDAR uses pulsed lasers to build a point cloud, which is then used to construct a 3D map or image. ToF applications create depth maps based on light detection, usually through a standard RGB camera. The advantage of

ToF over LiDAR is that ToF requires less specialised equipment so that it can be used with smaller and less expensive devices.

Unlike the previously mentioned single-sensor reconstruction approaches, the structure from motion technique does not technically use a single viewpoint [97]. The method uses a single camera, but this camera moves around in space, hence creating multiple virtual viewpoints and the camera position from each viewpoint is estimated from the images. The method can convert the observed images into a 3D reconstruction using point correspondences between different virtual camera views. Such a method has numerous applications in AR, i.e., placing an augmented object on a surface in a realistic fashion. However, when objects other than the camera are moving around in the observed environments, structure from motion runs into its inherent limitations because the point correspondences are no longer static. Since we primarily aim to reconstruct moving objects, this method has not been explored further.

To conclude, the main drawbacks of single-sensor reconstruction are the constraints imposed on the environment (shape from shading, structured light), the limitation to 2.5D reconstruction (stereo cameras, ToF cameras, LiDARs) and the inferior results when applied to moving objects (structure-from-motion). Because of these shortcomings, we will further investigate multi-sensor approaches.

4.1.2 Multi-sensor reconstruction

One obvious approach to mitigate the shortcoming of 2.5D reconstruction is to deploy multiple sensors with different viewpoints. Such a multi-sensor approach is not always feasible due to interference between sensors, e.g., structured light or the high cost of such a system, e.g., multiple LiDARs. For instance, one attempt has been made to reduce interference by inducing motion blur using a simple offset wave vibration motor [14]. Each sensor, therefore, only detected its own pattern sharply because both the projection of the structured light pattern and the camera are affected in the same way by the motor. In contrast, patterns of other cameras are perceived as blurry.

Multi-sensor approaches are capable of creating a full 3D reconstruction. Another advantage is that larger areas can be covered by adding more sensors. However, some sensors are not suitable for larger areas because of their limited range. For instance, ToF cameras only work reliably up to a few meters. RGB cameras do not have these limitations. Light reaches the sensor, even if it comes from far away. Therefore, the number

of RGB cameras needed to cover a large area is smaller compared to the number of ToF cameras to cover the same area, which makes an RGB camera setup relatively cheaper.

RGB cameras are used in this PhD, mainly due to their broad availability. We preferred to use several cameras positioned around the reconstruction volume with partly overlapping views instead of multiple stereo camera pairs for two different reasons. First, the processing speed of a stereo camera is slower compared to other multi-camera approaches due to stereo matching, which is computationally expensive. Second, when occlusion occurs, both cameras will not be able to capture the entire object because their viewpoint is almost identical. Having more different viewpoints enables more accurate 3D reconstructions because the occlusion can be appropriately handled, and this is the main focus of this work. For the same number of cameras in multiple stereo setups, only half of the viewpoints are possible compared to the proposed camera setup.

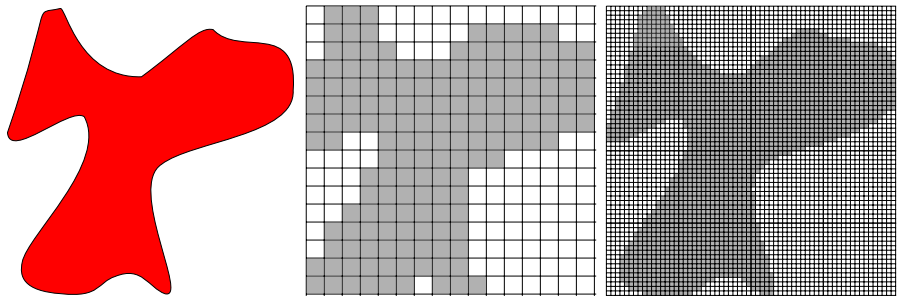
There are two common approaches for multi-camera setups with overlapping views to solve the 3D reconstruction: shape-from-silhouettes and space carving. The two methods differ in complexity because, with shape-from-silhouettes, only the silhouettes of an object are considered and reprojected into 3D space. In contrast, for space carving, colours from different views are also considered. In the remainder of this chapter, we will first discuss different ways to represent 3D objects because this choice impacts the implementation choices of shape-from-silhouettes. Space carving is also discussed at the end of this chapter.

4.2 Representation of 3D objects

Two significant choices can be made when it comes to representing 3D objects, similar to the representation of 2D images: voxel-based representations, which are similar to the pixel representation of images, and volumetric mesh-based representations, which are similar to vector images in 2D. In the following section we will discuss both in more detail.

4.2.1 Voxel-based representations

Voxel-based representations are discrete representations of a shape, similar to how silhouettes in an image are represented by pixels. A voxel is the smallest volumetric entity in such a representation. The most straightforward way to subdivide a reconstruction volume, is by using



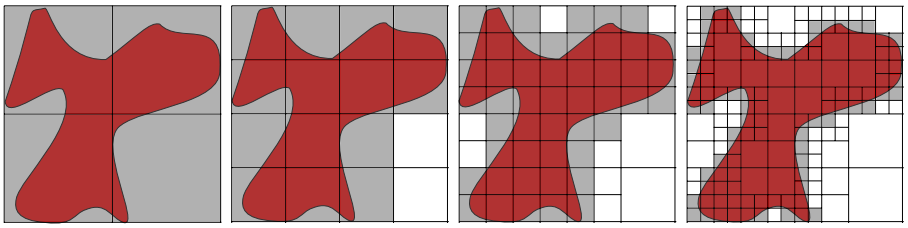
(a) object *(b) large voxel size* *(c) small voxel size*

Figure 4.1: *Fixed voxel grid representation (one slice in 3D). (a) The input, (b) and (c) are the voxelized reconstructions represented in grey (occupied) and white (unoccupied). A smaller voxel size can capture details more precisely.*

equally-sized voxels. Each voxel is defined by its width s_w, length s_l and height s_h and its position (s_x, s_y, s_z) on the voxel grid. Voxels do not overlap. Therefore, the voxel grid is also defined by the voxel's dimensions. The number of voxels to represent a particular volume is inversely proportional to the chosen dimensions of a voxel. Smaller voxels allow finer detail but require more memory. The memory requirement should not exceed the computers available memory.

A voxel-based 3D reconstruction is defined as the set of voxels that are occupied in the voxel grid. Voxels that are not part of the 3D reconstruction are unoccupied. Figure 4.1 shows the representation of one slice of the reconstruction of an arbitrary shape. A voxel coloured in white represents one that is unoccupied (the voxel is not part of the reconstruction). A voxel coloured in grey means that the voxel is occupied and, therefore, part of the reconstruction. The smaller the voxel size, the more precise the reconstruction becomes. Discretisation will be discussed in Section 4.3.

As mentioned before, fixed voxel grid representations may require a large amount of memory. Therefore, another representation is often used: octrees. Octrees can be seen as hierarchical multi-scale voxels. The reconstruction process is simple. An octree reconstruction initialises with a node that represents the entire reconstruction volume. In each iteration, nodes that are not completely occupied or unoccupied are further divided into eight nodes, half of the size in each direction compared to the parent node, whereas unoccupied nodes are discarded. This iterative procedure continues until all remaining nodes are occupied or when the desired minimum voxel size is reached. Figure 4.2 represents the octree iteration of a single slice in de 3D space where octrees are split



(a) iteration 1 *(b) iteration 2* *(c) iteration 3* *(d) iteration 4*

Figure 4.2: *Octree-based reconstruction: only nodes that are not entirely occupied are divided in the next iteration. Gray octrees are occupied, white octrees are unoccupied. The result is a representation of on object with fewer entities of different size, which is more memory-efficient compared to a voxel-based representation.*

in 4 in each iteration. The result of an octree-based representation is one where the building blocks may not all have the same size anymore.

Comparing Figure 4.1 and 4.2b learns us that instead of 16x16 = 256 small voxels that need to be checked, in this case 272 nodes are evaluated and subdivided (4, 16, 56, 196 per octree level). If there is more unoccupied space, the difference between the octree-based method and the standard voxel-based method becomes larger. The object is represented with 110 nodes with octrees instead of 159 fixed-size voxels. The reconstructed volume is identical for both an octree-based or voxel-based representation when the minimum voxel grid is equal to the fixed-sized voxel grid.

4.2.2 Volumetric mesh-based representations

Volumetric mesh-based representations consist of vertices and faces. A 3D simplex is a tetrahedron, a 3D volume with four vertices and four faces, of which each face is a triangle. Therefore, the union of different tetrahedrons can also represent a 3D reconstruction. If only the surface of a 3D object is represented, the reconstruction can be limited to a set of 3D triangles. Figure 4.3 shows an example of a bunny volumetric mesh with different numbers of vertices and faces (level of detail) to describe the same model. The most detailed representation has low errors but is less suitable for analysis. However, a significantly higher number of tetrahedrons need to be evaluated compared to a model with fewer vertices and faces.

Meshes can represent 3D objects very accurately and more smoothly compared to voxel-based representations because the vertices and faces

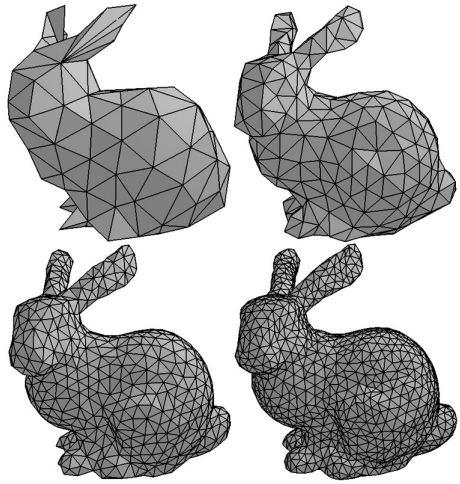


Figure 4.3: *Volumetric mesh reconstruction at different quantization levels (Source: cathyatseneca.gitbooks.io). A mesh reconstruction can be very detailed because the vertices of such a model are not limited to lie on a fixed grid. Mesh models are typically scalable and widely used in virtual environments such as AR/VR applications.*

are not bound to a predefined grid. However, the calculation of accurate meshes is time-consuming. In order to perform real-time mesh reconstructions, some optimisations are used, which usually have an impact on the mesh accuracy due to significant quantisation errors.

Note that there are also algorithms available to convert a fixed voxel-size representation to a mesh-based representation e.g., using the marching-cubes algorithm [59] in a single pass. However, such a conversion only approximates the best possible mesh of that object because the vertices of a mesh obtained with the marching-cubes algorithm do not necessarily lie on the surface of the reconstructed object.

4.2.3 Discussion: which representation fits best?

Each of the representation discussed above has its advantages and disadvantages. While voxel-based representations may require a large amount of memory, these representations are more suitable for further analysis, such as interior/exterior testing and volume overlap detection (e.g., preventing collisions). Voxel-based reconstruction is a massively parallel operation that allows multi-threading. Octree-based representations are especially suitable when the available memory is limited. However,

they require additional calculations due to their hierarchical complexity compared to voxel-based reconstructions.

Meshes can represent a surface in more detail, and the surface points are not restricted to that of a predefined voxel grid. Mesh models can also easily be scaled, making them suitable for AR/VR and with minor artefacts compared to scaled versions of a voxel-based representation. Meshes also require significantly less memory to be stored because the number of vertices and faces are usually much lower than the number of voxels. However, mesh-based representations require much more calculations to test if a point lies inside or outside the object, compared to a simple occupancy lookup in the voxel-based representation.

Since in this work we are mostly interested in fast reconstruction and further analysis where interior/exterior testing is essential, we resort to voxel-based reconstructions rather than photorealistic representations of the objects.

4.3 Shape-from-silhouettes

Shape-from-silhouettes is a concept introduced by Laurentini in 1994 [52]. The method describes how the backprojection of 2D silhouettes from a certain object can be used to obtain an approximated 3D shape of that object (Figure 4.4 and 4.5). The technique allows the reconstruction of arbitrary objects with a limited number of cameras placed around them. Laurentini presented his work for image sets only (spatial reconstruction). Due to the increase of computing power, it is now possible to apply the same technique in a fraction of a second, enabling application in the spatio-temporal domain with moving objects such as runners and cars in traffic. An important condition is that all frames from different cameras need to be synchronised.

We will now define the method more formally. First, we start with the difference between the shape-from-silhouettes and the visual hull. For an object $\mathcal{S} \subset \mathbb{R}^3$, the visual hull depends on the object $\mathcal{S}$ and the viewing set $\mathcal{V} = \{\mathrm{V}_1, \ldots, \mathrm{V}_N\}$, where each of the N viewpoints V_j corresponds to a camera position (j indicates the index of a camera), more specifically the camera centre.

Definition 1. *The visual hull* $\boldsymbol{VH}(\mathcal{S}, \mathcal{V})$ *of an object* $\mathcal{S}$ *relative to a viewing set* $\mathcal{V}$ *is a region of* $\mathbb{R}^3$ *such that, for each point* $P \in \boldsymbol{VH}(\mathcal{S}, \mathcal{V})$ *the half-line starting at* V_j *and passing through* P *contains at least one point of* $\mathcal{S}$ *for each viewpoint* $V_j \in \mathcal{V}$*. (Definition from [52]).*

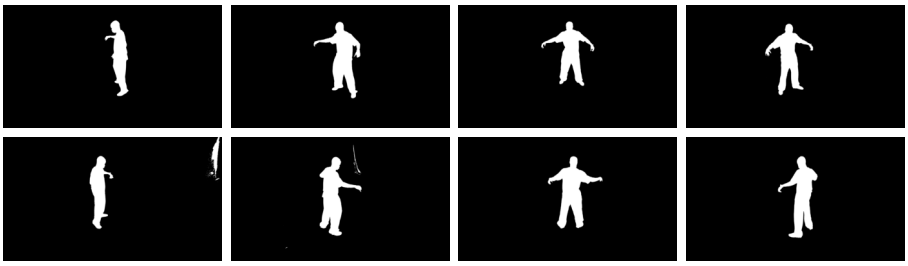


Figure 4.4: *A number of synchronized silhouettes from 8 calibrated cameras (CVSSP dataset [93]). The segmentation is very reliable because the breakdancer was captured in a green key studio.*

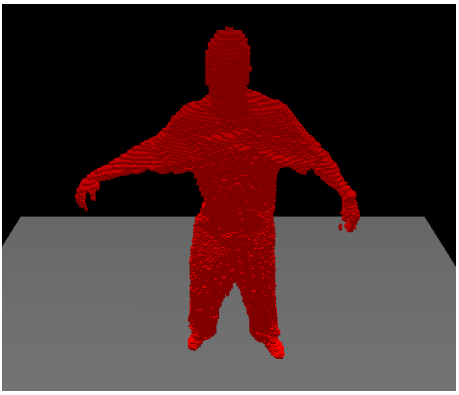


Figure 4.5: *Voxel-based shape-from-silhouettes reconstruction. This result is generated by using the input silhouettes in Figure 4.4. The voxel size is 10 mm, which shows considerable detail, e.g., near the hands of the person.*

Hence the visual hull is a geometric object that depends on the shape and position of the 3D object and the camera centres V_j. In computer vision, we usually do not know the geometry of the 3D object. Each camera can only observe a projection of the object (the silhouette, see Chapter 3). These silhouettes may not be equal to the projection of the 3D object due to errors in de segmentation or occlusion. In the most general form, a silhouette is a region of the 2D image plane. We define a more practical definition of shape-from-silhouettes as follows.

Definition 2. *The shape-from-silhouettes* $\boldsymbol{SfS}(I, \mathcal{V})$ *of the observed silhouettes* I_j *from a set of viewpoints* $\mathcal{V}$*, is the set of all points* $P \in \mathbb{R}^3$ *such that, for each viewpoint* $V_j \in \mathcal{V}$*, the projection of* P *on the image plane of camera* j*,* P_j *is part of silhouette* I_j*.*

Note that both the image resolutions and the reconstruction volume are quantised and cause inaccuracies in the reconstruction. Smaller voxel

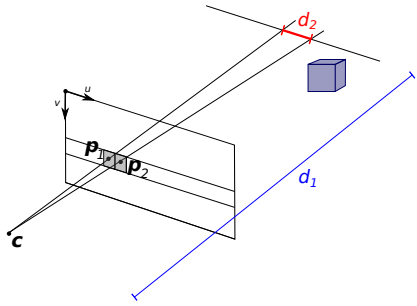
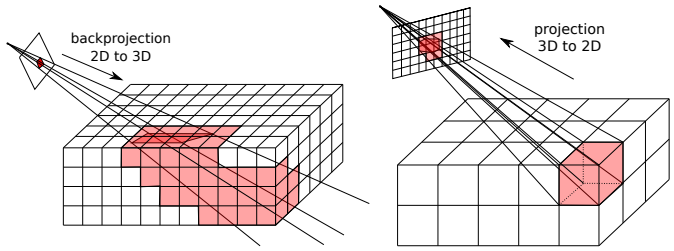


Figure 4.6: *Relation between pixel size and voxel size. Maximum precision depends on the distance d_2 between the backprojected lines CP_1 and CP_2 at distance d_1. An appropriate voxel size has the same magnitude as the distance d_2.*

sizes correspond to more detailed voxel reconstructions, but the image resolution limits the detail. The minimum voxel size depends on the image resolution and the distance between the camera and the object. A suitable voxel size can be determined by measuring the distance between the backprojected pixel centres of two neighbouring pixels at the desired distance (location of the object) in front of the camera. This distance mainly depends on the focal length of the camera and the lens distortion. Figure 4.6 illustrates this the relation between two pixel p_1 and p_2 and the distance between them (d_2) at distance d_1 in front of the camera. The cuboid shows the size of a voxel where each side is d_2 long. The smallest size used for a voxel reconstruction should not be smaller than half the size of this cuboid in each dimension, related to the Nyquist Theorem for sampling.

The goal of shape-from-silhouettes is to obtain a shape, which equals $\mathbf{VH}(\mathcal{S}, \mathcal{V})$. However, the use of silhouettes introduces an additional challenge to accurately reconstruct the 3D object because each silhouette is prone to errors. Errors may be due to bad silhouette detections. However, in many cases, they are due to other objects blocking the view of the object of interest, either partially or completely, from a viewpoint V_j, the so-called occluders. As mentioned above, the 2D image plane is quantised into pixels, so a silhouette becomes a finite set of pixels, which approximates the real silhouette. However, these quantization errors are relatively small for high-resolution images.

In this chapter, we focus on 3D reconstruction under the assumption that the silhouettes are more or less complete (no occlusion, silhouette F1-score of 90% or above, see Section 3.4). We refer to Chapter 5 for our solution for incomplete silhouettes where occlusion is appropriately



(a) backprojection of pixels *(b) voxel's projection*

Figure 4.7: *Voxel-based reconstruction using (a) silhouette backprojection or (b) voxel projection. Intersecting voxels and pixels are coloured.*

handled.

In the remainder of this chapter, we will discuss several choices and properties of the shape-from-silhouettes algorithm in terms of implementation and data structures of state-of-the-art methods.

In Section 4.2 we discussed voxel-based and mesh-based representations of 3D objects. This section will discuss two families of algorithms: one to generate a voxel-based 3D reconstruction and another one to generate a mesh-based 3D reconstruction. Note that we are looking to perform the shape-from-silhouettes process in real-time. Therefore, the choices we make are influenced by processing time, especially when there are different ways to obtain similar results.

4.3.1 Voxel-based shape-from-silhouettes

Voxel-based shape-from-silhouettes is a simple and straightforward approach to approximate a 3D object from a set of observed silhouettes and their corresponding camera calibration. There are two strategies to reconstruct the shape of an object based on silhouettes using voxels.

1. **SfS based on silhouette backprojection**
   Initialise a counter to 0 for each voxel. Backproject the silhouettes and increment the counter for all voxels inside the reprojected silhouette. The final reconstruction equals the set of voxels for which its counter is equal to the number of cameras.

2. **SfS based on voxel projection**
   Visit all voxels in the reconstruction volume and calculate the voxel's projection on each of the image planes. The final reconstruction is the voxel set, for which the voxels' projection overlaps with the silhouette in all camera views.

Data: camera calibration (N cameras), silhouettes I_j, the set of all voxels in the reconstruction volume $\mathbb{S}$
Data: voxel-based shape-from-silhouettes reconstruction Ω
Procedure:
$n_v = 0$ for all $v \in \mathbb{S}$
for $j \leftarrow 1$ *to* N **do**
  T = findcontours(I_j)
  foreach contour t in T **do**
    foreach v *in* $\Pi_j^{-1}(t)$ **do**
      $n_v = n_v + 1$
    end
  end
end
$\Omega = \bigcup v$ for all $v \in \mathbb{S}$ where $n_v = N$

Algorithm 4.1: *SfS based on silhouette backprojection. Each contour t of each silhouette I_j per camera j is backprojected. The set of voxels contained by the intersection of all backprojections forms the voxel reconstruction Ω.*

Figure 4.7a shows how a pixel is backprojected and indicates all voxels with which the backprojection intersects. In Figure 4.7b, we project a voxel onto the image plane and indicate which pixels cover its projection. The voxel grid size was adapted for visualisation purposes, so it is clear that a voxel's projection covers multiple pixels, and the backprojection of a pixel covers multiple voxels.

Both algorithms can be expressed in a few lines of code with different implementations. However, we chose to illustrate how the algorithm works by not over-optimising.

Algorithm 4.1 shows the first approach. In essence, the borders of each of the silhouettes I_j is searched. This process has complexity $\mathcal{O}(I_w I_h)$ with $I_w I_h$ the total number of pixels in the image. Each of these contours t is backprojected: $\Pi_j^{-1}(t)$. Such a backprojection forms a generalised cone, having the viewpoint V_j as a top and its surface defined by the contour points t. The count n_v of all voxels $v \in \mathbb{S}$ which also lie inside this generalised cone are incremented by 1. After evaluating all silhouettes from all views, the union of voxels $v \in \mathbb{S}$ where $n_v = N$ represents the voxel-based visual hull Ω.

Algorithm 4.2 loops over all voxels v in the reconstruction volume and increments a when the projection of voxel v_j lies inside silhouette I_j for each view. If $a = N$, the voxel projects inside the silhouette in each camera. Therefore, the voxel becomes part of the reconstruction Ω. This

Data: camera calibration (N cameras), silhouettes I_j, the set of all voxels in the reconstruction volume $\mathbb{S}$
Result: voxel-based shape-from-silhouettes reconstruction Ω
Procedure:

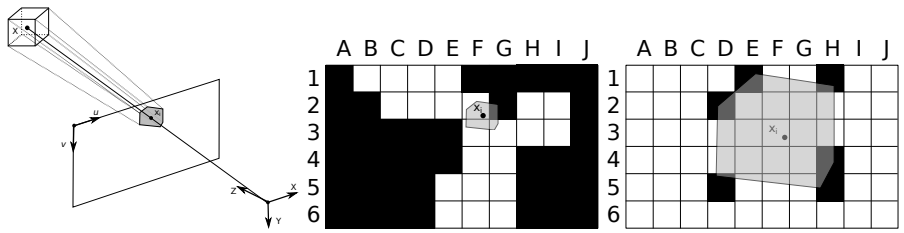
```
Ω = ∅
foreach v ∈ 𝕊 do
    a = 0
    for j ← 1 to N do
        if v_j ⊂ I_j then
            a = a + 1
        end
    end
    if a = N then
        Ω = v ∪ Ω
    end
end
```

Algorithm 4.2: *SfS based on voxel projection. Each voxel v in the reconstruction volume $\mathbb{S}$ is projected onto the image j and compared with the silhouette I_j. If the voxel projects inside each silhouette, the voxel is occupied. The set of occupied voxels forms the voxel reconstruction Ω.*

algorithm has complexity $\mathcal{O}(N_v)$ where N_v the total number of voxels in the reconstruction volume.

Both algorithms produce the same voxel-based reconstruction. However, there are many reasons why the second algorithm is used in this work. First, the computation time does not depend on the size and complexity of the silhouettes but has a constant computation time because every voxel is visited exactly once. Since we mainly work on live applications, it is essential to optimise this computation time. We found that using a look-up table for the voxel projections is faster than recalculating these projections over and over again. In theory, the same can be done for the first approach, but to store all voxels inside a pixel's backprojection requires a significant amount of memory because the number of voxels inside a pixel's backprojection (approach a) is much larger than the number of pixels inside a voxel's projection (approach b).

The voxel's projection is a convex polygon with at most six vertices. For all pixels inside this convex polygon, we should verify if they are part of the silhouette I_j (full voxel projection test). However, if the voxel's projection overlaps with a few pixels only (small voxel size), we noticed that it suffices only to check if the voxel's centre projects inside the silhouette I_j (= single pixel projection test)). Suppose a voxel's



(a) *(b)* *(c)*

Figure 4.8: *Projection test optimization: instead of testing the entire grey area of the projected voxel, only the projected centre x_i needs to be checked. White pixels represent the silhouette.*

projection covers a large number of pixels. In that case, this optimisation should not be used, and another criterion can be chosen to avoid checking all pixels inside the convex polygon of a voxel, e.g., only its vertices. The criterion may also allow a pixel to not fully overlap with a voxel to avoid the reconstructed object becoming too large. Figure 4.8 illustrates a good (b) and bad example (c) of when this optimization is useful. For the first approach, a comparable optimisation can be thought of as having the number of voxels in a pixel backprojection equal to the line through the voxel volume. That, however, requires a small voxel size, and this optimisation can only be used at that voxel size (or even smaller, which is not efficient). Otherwise, some voxels would not be marked as occupied and holes in the reconstruction would be created. See Section 4.5.2 for numerical results about this optimization.

4.3.2 Mesh-based shape-from-silhouettes

An alternative approach to voxel-based reconstruction is calculating the intersection points of reprojected silhouette borders. The camera position defines the corresponding generalised cones, and the silhouette border defines the shape of this polygon (Figure 4.9). These intersections can be represented as a mesh. These mesh-based solutions are not bound to a discretised voxel grid. Therefore, mesh-based solutions can deliver more precise shape reconstructions compared to voxel-based approaches. However, current methods have not obtained the same processing speed compared to voxel-based approaches, making it unsuited for real-time applications.

Researchers have been investigating mesh-based reconstructions for a while now. In 2001, Matusik et al. reported 15 fps on a 2x933MHz Pentium III PC, where each camera image is first processed on a separate

600 MHz Athlon desktop PC [62], which can be considered near real-time. However, back then, they could only obtain this processing speed with input images at a reduced resolution of only 320×240 pixels and only for four camera views. In order to maintain a relatively constant frame rate, the input silhouettes are simplified with a coarser polygonal approximation. This approximation omits the biggest advantage of mesh-based reconstruction by giving up even more precision after the resolution was already reduced. On a modern computer, this algorithm would most likely run even faster. However, increasing the number of cameras and the image resolution may again drop the gain in computation time.

Sablatnig et al. [26] proposed an approach to reduce the time required to build a shape reconstruction by reducing the number of input views by taking into account object features to ensure a certain level of model accuracy. Note that his method was developed for turntable setups. The method compares subsequent silhouettes from different turntable positions and picks those silhouettes that are significantly different according to a threshold value. The paper presents an algorithm for next-view planning with a minimal number of different views.

Yemez et al. [111] also presented a system for a mesh-based approach with high-resolution shape and texture. All aspects of such a system are discussed, from the camera calibration to the generation of a triangular mesh. Although the result of this approach is a mesh, the algorithm itself uses an octree-based approach, and therefore, the surface points are impacted by quantisation. This issue is solved using the marching-cubes algorithm and an interpolation algorithm to create a smoother surface. The visual result for static objects look promising, but nothing is mentioned about computation time. Since only static objects are being reconstructed, the method most likely cannot process images in real-time.

Miline [66] developed a modified marching-cubes method that can quickly compute an object's volume from its visual hull by using multiple views of the object. In this method, the first step is the voxelisation of the volume containing the target object. Then, the marching-cubes algorithm is used to approximate the surface passing through the exterior voxels. Finally, the positional accuracy of the surface is improved using a binary search. Miline claimed that by applying the marching-cubes algorithm to a low-resolution voxel-based model (instead of a high-resolution model), better results could be achieved in a shorter time. The binary search can further improve the marching-cubes model with a minimal computational expense.

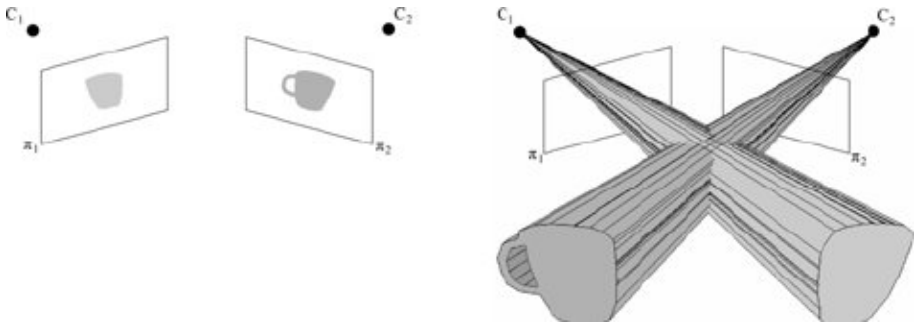


Figure 4.9: *Intersection of backprojected general cones reconstruction [63].*

More recently, Phothong et al. [74] proposed an algorithm in 2017 that uses the direct intersection of multiple sets of the generalised cones. The surface points are triangulated according to the topological relationship of the obtained points and the polygons corresponding to each of the points. This topology is exploited using strategies similar to octrees. The main advantages of the proposed method are that all 3D points are precisely located on the silhouettes of 2D images. However, the processing speed is limited to 0.5-1.3 seconds for 16 camera views on a computer with a 1.90 GHz CPU and 4 GB of RAM.

Although the mesh-based reconstruction methods may generate more precise 3D reconstructions, the calculations are more complex and require more computation time than voxel-based approaches. Moreover, voxel-based approaches consist of many processes that can be executed in parallel.

4.4 Space carving

An alternative approach to the shape-from-silhouettes technique is called space carving [50, 18]. This method considers the colour of the object to create the photo hull of an object. A photo hull is different from a visual hull because its volume is equal to or smaller than a visual hull and all voxels in a photo hull are photo-consistent. The technique relies on the fact that if a point is part of the object's surface, the colour of that point should be similar in all camera views, observing this point. Photo-inconsistent voxels can hence be removed. We note that this assumption only holds true for Lambertian reflectance where similar colours can be observed from different viewpoints.

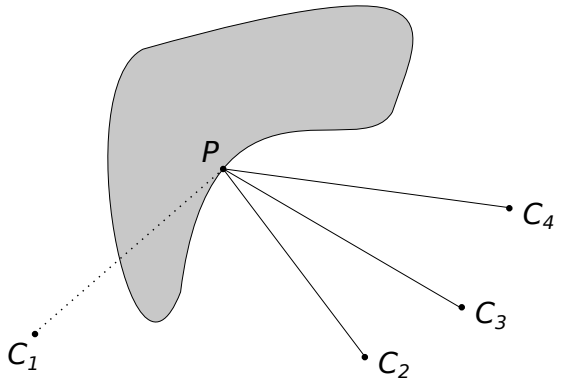


Figure 4.10: *Point P lies in the direct line of sight of cameras C_2, C_3, C_4 whereas another part of the grey object of interest is occluding the line of sight of camera C_1, therefore, camera C_1 should not be taken into account for photo-consistency as the observed part of the object might have a different colour than point P in camera C_1.*

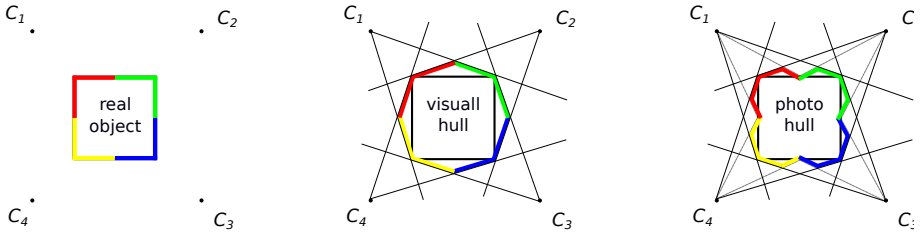


Figure 4.11: *Visual hull vs. photo hull. The colour consistency helps to carve inside the visual hull and hence create a smaller shape, which is still larger than the object.*

The technique is much more complex than shape-from-silhouettes. It uses colour information. An analysis is needed to check the visibility of that point to decide on the so-called photo-consistency of a point's appearance in different images. Figure 4.10 illustrates the self-occlusion problem. Camera C_1 is unable to observe point P because part of the object of interest prevents direct line of sight. Therefore, the colour of the corresponding pixel on the image sensor of camera C_1 might be different from that of point P and camera C_1 should not be used in assessing photo-consistency of the voxel at location P. Due to the inclusion of photo-consistency, it becomes possible to carve away voxels inside the visual hull of the object.

Figure 4.11 shows how the colour is used to make the reconstructed object smaller than the visual hull. A smaller volume means that this shape is closer to the object's actual shape if the colour matching is

Data: camera calibration (N cameras), colour input images
Data: voxel-based photo hull reconstruction $\mathcal{V}^*$
Procedure:
Step 1: Initialize $\mathcal{V}$ to shape-from-silhouettes.

Step 2: Repeat the following steps for unchecked voxels $v \in \text{Surface}(\mathcal{V})$ until a non-photo-consistent voxel is found:

a. Project v to all photographs in $\text{Visible}_j(v)$. Let $\text{col}_1, \ldots, \text{col}_j$ be the pixel colours to which v projects and let $\zeta_1, \ldots, \zeta_j$ be the optical rays connecting v to the corresponding optical centres.

b. Determine the photo-consistency of v using $\text{consist}_K(\text{col}_1, ..., \text{col}_j, \zeta_1, \ldots, \zeta_j)$.

Step 3: If no non-photo-consistent voxel is found, set $\mathcal{V}^* = \mathcal{V}$ and terminate. Otherwise, set $\mathcal{V} = \mathcal{V} - v$ and repeat Step 2.

Algorithm 4.3: *Space carving algorithm. Starting from the shape-from-silhouettes, this method removes photo-inconsistent voxels from the reconstruction. A voxel is photo-inconsistent if the colours inside its projection in the images are not similar, at least not for the cameras that can observe the voxel. The check consists of a visibility check and a colour-consistency check.*

performed correctly. An analysis of shape reconstructions closer to the actual shape will produce more reliable results in general.

Algorithm 4.3 shows how the photo hull can be obtained. An important step to understanding the complexity of the algorithm is the function $\text{Visible}_j(v)$ which determines for a camera j if a voxel v is in its direct line of sight. More specifically, there is no (self-)occlusion. Cameras for which the optical ray ζ_j to a voxel v passes through other occupied voxels, are excluded from $\mathcal{V}$ (e.g., camera C_1 in Figure 4.10). The number of operations to find these voxels depends on the size of the reconstruction volume and the number of cameras. For the K cameras of which the voxel can be observed, the colour is compared to determine if the voxel's colour is photo-consistent in all camera views.

The consist_K function is used to compare colours. It represents an algorithm that takes as input at least $K \leq N$ colours $\text{col}_1, \ldots, \text{col}_K$, K vectors $\zeta_1, \ldots, \zeta_K$, and the light source positions and decides whether it is possible for a single surface point to reflect light of colour col_j in direction ζ_i, simultaneously for all $j = 1, \ldots, K$. Note that such a comparison can only be performed reliably when the light sources can

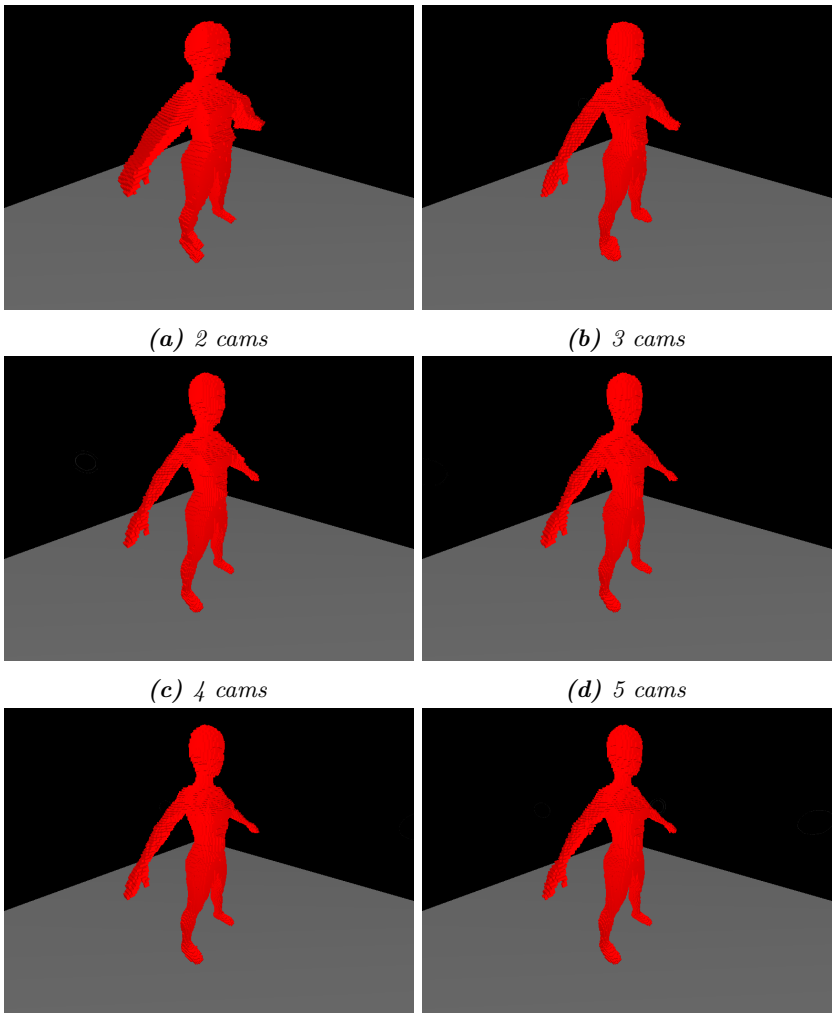
Figure 4.12: *Reconstruction of a breakdancer using space carving. Each surface voxel is coloured with photo-consistent colour.*

be modelled (controlled indoor environments). We refer to Chapter 3 for suitable colour spaces.

Figure 4.12 shows an example of a breakdancer captured with eight cameras in a green screen studio to facilitate the segmentation. The texture of the breakdancer has been added to the reconstructed model the generate a photorealistic result. We have not explored this method further because of its increased computation time compared to the standard shape-from-silhouettes implementation.

4.5 Shape reconstruction experiments

This section shows several experiments related to volumetric shape reconstruction. A voxel-based volumetric shape reconstruction depends on a number of variables: the position of the cameras, the number of cameras, the complexity of the shape and the voxel size. For a static object of which the shape is more or less known, an ideal camera configuration could be theoretically achieved. However, since the orientation of the object impacts the optimal camera configuration directly, we do not optimize for particular poses and use the rules of thumb that were explained in Section 2.2.



(a) 2 cams *(b) 3 cams*

(c) 4 cams *(d) 5 cams*

(e) 6 cams *(f) 7 cams*

Figure 4.13: *The volumetric shape reconstruction of the body's shape using 2 to 7 cameras. In general, the reconstructions is more precise with more cameras but that is only true when the camera positions remain the same which is not the case in this experiment.*

We simulate the results by using a 3D model of a person and projecting the model on each of the camera views to create the silhouettes. Using these silhouettes and the known camera positions, we can use the shape-from-silhouettes algorithm to reconstruct the person again in 3D, and hence, we can compare the volumetric shape reconstruction with the model to determine the accuracy.

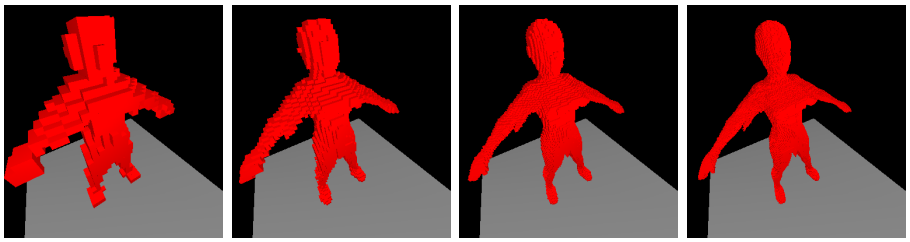
4.5.1 Camera configuration

This experiment also shows the impact of the number of cameras on the 3D reconstruction. The person is standing in a starfish pose and the cameras are placed on a circle with 3 m radius around the person, 80 cm from the ground plane and equidistant from each other in case of an odd number of cameras. To avoid that cameras are placed opposite of each other, the angles for half of the cameras in an even setup is shifted with $\frac{\pi}{N}$ for half of the cameras. The number of cameras varies between 2 and 7. A top view camera was also considered but because the arms occlude the sides of the person, the top view camera was not a useful addition to improve the reconstruction, so we will not further consider it.

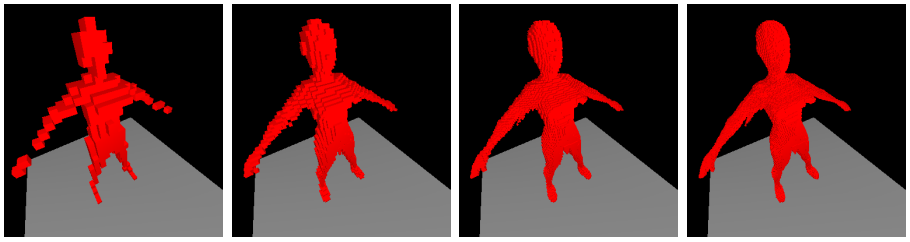
Figure 4.13 shows the impact on the reconstruction of the number of cameras in the camera configuration. A clear excess of voxels can be seen when only 2 of 3 views are used. The more cameras, the tighter the 3D reconstruction becomes around the actual model. In general, more cameras improve the reconstruction but also increase the cost of such a system. It is best to determine the impact of extra cameras compared to the gain they bring. At a certain point only marginal gain (or none at all) is reached. In that case, it makes little sense to increase the number of cameras further.

4.5.2 Voxel size

Figure 4.14 shows visually how the shape reconstruction is influenced by the voxel size. We consider two different projection test during the reconstruction. In the top row, the voxel is projected (FVPT = full voxel projection test) on the image sensor of each cameras, which corresponds to the pixels in the image of the convex hull of the projected vertices of the voxel. A voxel is occupied when at least one pixel is part of the voxel's projection in each view. The bottom row shows the results if only the the single pixel corresponding to the voxel centre's projection



(a) *50 mm (FVPT)* ***(b)*** *20 mm (FVPT)* ***(c)*** *10 mm (FVPT)* ***(d)*** *5 mm (FVPT)*



(e) *50 mm (SPPT)* ***(f)*** *20 mm (SPPT)* ***(g)*** *10 mm (SPPT)* ***(h)*** *5 mm (SPPT)*

Figure 4.14: *The figures show the volumetric shape reconstruction of the body's shape using five cameras and a voxel size between 5 and 50 mm. A voxel size of 50 mm is not adequate for a detailed reconstruction of a person. The top row show the results of the full voxel projection test (FVPT) and the bottom row show the results of the single pixel projection test (SPPT).*

is evaluated (SPPT = single pixel projection test). In general, small voxel sizes lead to a detail reconstruction than large voxel sizes.

The full voxel evaluation approach results in a thicker reconstruction, while the optimized reconstruction tends to be slimmer. When voxel sizes become smaller, the apparent differences between both methods disappear. It is important to note that the optimization should not be used when the voxel size is comparable to the order of magnitude of the shape. For instance, the wrists are missing in the reconstruction of 5 cm because that is about the size of the wrist in this model.

We provide a numerical analysis in Table 4.1. To compare the estimated shape with the ground truth (a mesh model), we converted the mesh model to a voxel representation having the same voxel size. Voxel-to-voxel comparison was done where a label is assigned to each voxel: TP (true positive), FP (false positive), TN (true negative) and FN (false negative). The number of voxels in each class are used to calculate the precision ($\frac{\mathrm{TP}}{\mathrm{TP+FP}}$), recall ($\frac{\mathrm{TP}}{\mathrm{TP+FN}}$) and F1 score ($\frac{\mathrm{TP}}{\mathrm{TP}+\frac{\mathrm{FN+FP}}{2}}$).

For the full voxel projection implementation, recall is about 100%, and precision increases with decreasing voxel size. The optimised version

voxel size	precision	recall	F1	fps
50 mm (full)	67.09%	100.00%	80.31%	11.1
50 mm (opt)	96.61%	58.76%	73.08%	530
20 mm (full)	72.64%	100.00%	84.15%	1.6
20 mm (opt)	94.80%	87.42%	90.96%	297
10 mm (full)	76.92%	100.00%	86.95%	0.23
10 mm (opt)	91.54%	96.12%	93.77%	42
5 mm (full)	80.03%	100.00%	88.91%	0.02
5 mm (opt)	88.49%	99.13%	93.51%	6.2

Table 4.1: *Numerical analysis of voxel sizes between 5 mm and 50 mm in terms of precision, recall, F1-score and frame rate. Small voxel sizes mean slower reconstructions but better reconstruction in terms of F1-score. We reach higher frame rates by using the optimized projection test. However, this test also has an impact on the precision and recall. It should only be used with voxel sizes lower or equal than 20 mm.*

shows a different trend. Recall increases, and precision decreases with decreasing voxel size. This behaviour can be explained because mainly the voxels at the border are missing from the reconstruction. However, due to the excess property of a shape-from-silhouettes, these surface voxels are likely not part of the actual object. Therefore, these effects impact both recall and precision. In the last column, we also show the frame rate of each implementation (using 4 CPUs). The optimised projection test increases the frame rate significantly. For real-time applications, such optimisation is required. Note that we use a look-up table (LUT) of all the projections to speed up the reconstruction process. The LUTs increase the processing speed by roughly 30%. However, they require a significant amount of memory. Too much, in fact, for the 5 mm voxel size reconstruction on our system. A LUT is also used for the optimised implementation, but the memory requirement is significantly lower because only a single point needs to be stored per voxel per camera.

Also other metrics are important: reducing the voxel size by half in each direction requires eight times more calculations. Also, note that the frame rate depends on the size of the reconstruction volume. For this experiment, the size of this reconstruction volume was 2 m $\times$ 2 m $\times$ 2 m.

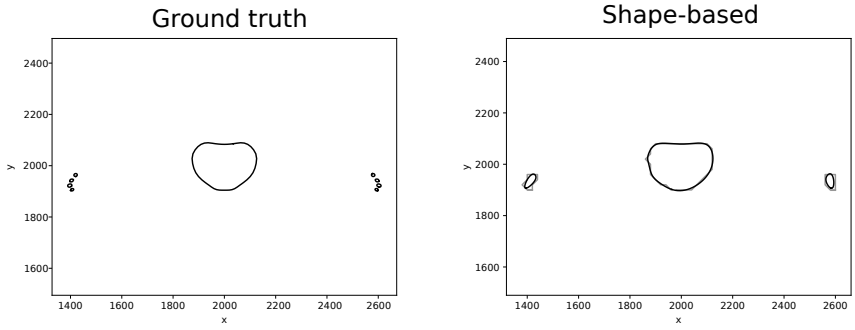
num cams	length (mm)	error (mm)
2 cams	1075	140
3 cams	965	30
4 cams	925	-10
5 cams	944	9
6 cams	925	-10
7 cams	923	-12

(a) *waist circumference (ground truth: 935 mm)*

num cams	length (mm)	error (mm)
2 cams	839	130
3 cams	756	47
4 cams	727	18
5 cams	720	11
6 cams	708	-1
7 cams	709	1

(b) *abdominal circumference (ground truth: 709 mm)*

Table 4.2: *Numerical results of the waist and abdominal circumference of a person using between 2 and 7 cameras.*



(a) *Ground truth* ***(b)*** *5 cams*

Figure 4.15: *Visual comparison of the abdominal circumference. Using 5 cameras we can accurately reconstruct a person's waist and measure the circumference. The extra shapes on the left and right of the waist are parts of the hands that are also reconstructed. They are ignored in the analysis.*

4.5.3 Shape analysis: garment fitting

The 3D reconstruction can directly be used for analysis. One of many application is to measure a person's size for garment fitting. We focus on the abdominal and waist circumference, but other sizes could also be measured. The person from the experiments above was reconstructed using a voxel size of 1 cm.

Each circumference is measured by analysing a slice. The outer voxels of that slice are considered as the surface voxels (skin) of the person. The quantisation error that comes with the voxelized 3D reconstruction is handled by estimating the voxel contour with a spline. Figure 4.15 shows the result of the abdominal circumference. The shapes are similar.

The shapes on the left and right side represent the fingers. They lay in the same slice but are not considered in the analysis of the waist.

Table 4.2 shows the numerical results of the analysis. We notice that for this particular analysis 4 or 5 cameras are sufficient to estimate the circumferences accurately with an expected error of 1 cm.

4.6 Conclusion

This chapter explained how 3D shape reconstruction could be performed from silhouette images in a calibrated camera setup. We assumed more or less complete silhouettes e.g., the reconstructed object is the intersection of all backprojected silhouettes. Under controlled circumstances, we can reliably reconstruct the objects in front of the cameras.

In Chapter 5 we will discuss the shape from incomplete silhouettes reconstruction. The incompleteness of the silhouettes may have to cause: poor silhouette extraction or occlusion. We will extend the shape-from-silhouettes reconstruction to generate reliable reconstruction based on incomplete silhouettes.

The presented shape reconstruction offers volumetric information of objects and humans. Some analysis can be performed on these shapes such as the garment fitting application. However, to analyse a person's pose in more detail, skeletal information is more suitable, which will be investigated in Chapter 6.

The main contributions made in this chapter are the following:

- real-time implementation of shape-from-silhouettes;
- development of a complete framework to calculate reconstructed object from input images given the camera calibration;
- analysis of important parameters in a voxel-based reconstruction: camera configuration and voxel size;
- basic analysis on the reconstructed voxel-based volumetric shape reconstruction.

Some of the work in this chapter is described in these publications:

- K. Bauters, J. Cottyn, D. Claeys, M. Slembrouck, P. Veelaert, and H. van Landeghem. Automated work cycle classification and performance measurement for manual work stations. *Robotics and computer-aided manufacturing*, 51, 2018. ISSN 0736-5845

- M. Slembrouck, D. Van Cauwelaert, P. Veelaert, and W. Philips. Shapes-from-silhouettes based 3d reconstruction for athlete evaluation during exercising. In *Abstracts of Science and Engineering Conference on Sports Innovations (SECSI'16)*, page 2, 2016

- J. Li, M. Slembrouck, F. Deboeverie, A. M. Bernardos, J. A. Besada, P. Veelaert, H. Aghajan, W. Philips, and J. R Casar. A hybrid pose tracking approach for handheld augmented reality. In *Proceedings of the 9th International Conference on Distributed Smart Cameras (ICDSC'15)*, pages 7–12. ACM, 2015. ISBN 978-1-4503-3681-9

- J. Li, B. Goossens, M. Slembrouck, F. Deboeverie, P. Veelaert, H. Aghajan, W. Philips, and J. R Casar. Demo: a new 360-degree immersive game controller. In *Proceedings of the 9th International Conference on Distributed Smart Cameras (ICDSC'15)*, pages 201–202. ACM, 2015. ISBN 978-1-4503-3681-9

- X. Xie, D. Van Cauwelaert, M. Slembrouck, K. Bauters, J. Cottyn, D. Van Haerenborgh, H. Aghajan, P. Veelaert, and W. Philips. Abnormal work cycle detection based on dissimilarity measurement of trajectories. In *Proceedings of the 9th International Conference on Distributed Smart Cameras (ICDSC'15)*, page 6. ACM, 2015. ISBN 978-1-4503-3681-9

- X. Xie, J. De Vylder, D. Van Cauwelaert, P. Veelaert, W. Philips, H. Aghajan, M. Slembrouck, D. Van Haerenborgh, H. Van Landeghem, K. Bauters, J. Cottyn, and H. Vervaeke. Average track estimation of moving objects using ransac and dtw. In *Proceedings of the 8th International Conference on Distributed Smart Cameras (ICDSC'14)*, page 6. ACM, 2014. ISBN 978-1-4503-2925-5

- K. Bauters, H. Van Landeghem, M. Slembrouck, D. Van Cauwelaert, and D. Van Haerenborgh. An automated work cycle classification and disturbance detection tool for assembly line work stations. In *Proceedings of the International Conference on Informatics in Control, Automation and Robotics (ICINCO'14)*, volume 2, page 7, 2014. ISBN 9789897580406

- K. Bauters, H. Van Landeghem, D. Van Haerenborgh, M. Slembrouck, D. Van Cauwelaert, P. Veelaert, and W. Philips. Multicamera complexity assessment system for assembly line work stations. In *Proceedings of the European Concurrent Engineering Conference (ECEC'13)*, page 5, 2013

Chapter 5

Occlusion Handling

Occlusion presents a challenge for detecting objects in real-world environments. This phenomenon causes cameras to capture incomplete scene data and leads to errors in the analysis if this is not considered. Standard shape-from-silhouette methods assume no occlusion and generate inadequate 3D reconstructions. This chapter discusses how we solved the occlusion problem in a multi-camera network having overlapping views by extending the standard shape-from-silhouettes algorithm. First, we define occlusion, after which we give an overview of the solutions in literature to arrive at our proposed solution and experiments.

5.1 Occlusion

5.1.1 Definition and problem statement

Occlusion occurs when a camera cannot fully observe an object due to obstacles being present in the direct line of sight between the camera and the object. When an object is only partially visible, we call the phenomenon "partial occlusion". Occlusion leads to tracking loss and errors in a 3D reconstruction when the occlusion is not handled appropriately. Figure 5.1 illustrates an example of partial occlusion. In this case, we want to reconstruct the person in the scene. An obstacle prevents camera C_1 from observing the lower part of the person, while camera C_2 observes the entire person because the obstacle is positioned behind the person from its perspective. The overlapping area between the backprojected observed silhouettes does not include the entire person (= inferior 3D reconstruction).

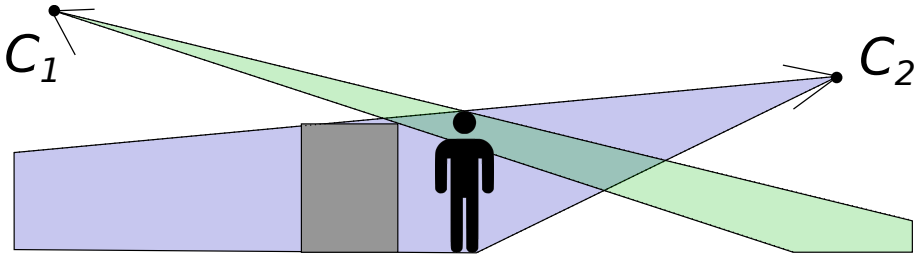
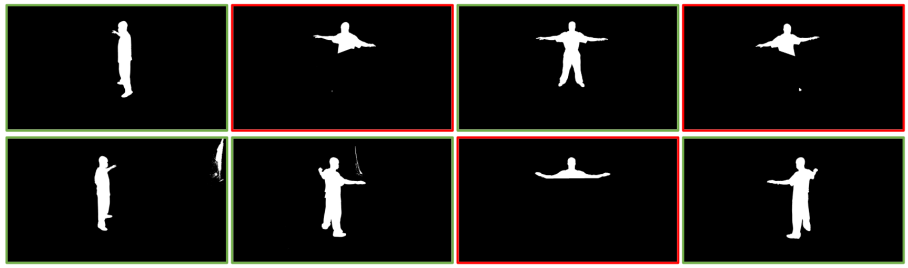


Figure 5.1: *This figure illustrates how occlusion happens. The person is partially occluded in camera C_1 by the obstacle, while camera C_2 can observe the entire person since the obstacle is positioned behind the person from its perspective.*

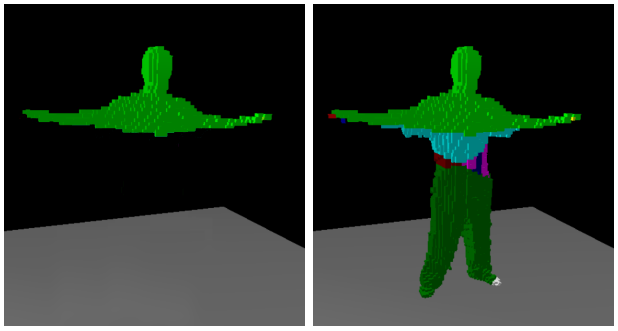
Occlusion is not the only reason why a camera may not observe an object completely. Real-world cameras have limited fields of view. Everything outside their field of view is also unobservable by that camera and limits the camera coverage. In a sense, the area outside the camera view can also be handled as static occlusion. As long as the camera does not move, the area out of view remains constant. The reasoning process can use the knowledge about occlusion from each camera to better understand the scene.

Scene understanding is vital to handle occlusion correctly. The goal of the 3D reconstruction process is to generate a reconstruction that is as small as possible while still including the complete object. By considering the locations of the occluders and the incomplete silhouettes, we can obtain this goal. Given enough cameras, the reconstruction can still become detailed enough for further analysis. For instance, in the example of Figure 5.2, the proposed reconstruction can reconstruct the complete person, even when three of the eight views are occluded.

We choose to reconstruct moving objects in the scene rather than only detect their position to offer more information to such a system. We find typical applications in traffic where different brands and sizes of vehicles exist. Moreover, their appearance may significantly change when attributes are added, such as trailers or bike racks. Even if a system can recognize a specific type of car, we risk underestimating the actual size of the vehicle. Our volumetric reconstruction algorithm uses no prior knowledge to reconstruct the object in the scene and handle both dynamic and static occlusion. A full 3D reconstruction can help to assess safety margins between objects better.



(a) input silhouettes from 8 cameras



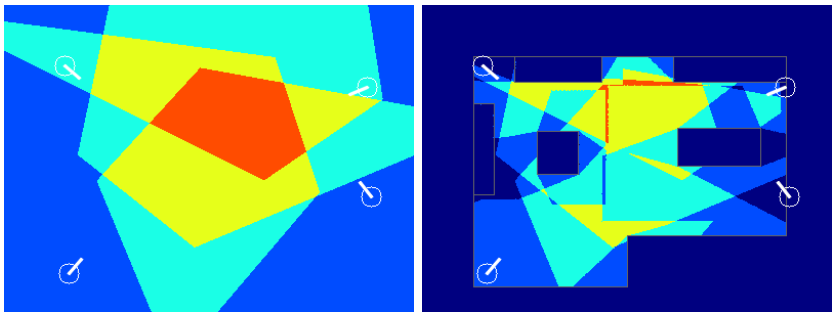
(b) shape-from-silhouettes *(c) proposed*

Figure 5.2: *3D reconstruction example of eight silhouettes, three of which are incomplete. In (b) the shape-from-silhouettes (SfS) which misses a large part of the reconstruction (occluded region) and in (c) our proposed reconstruction based on the same silhouettes. Different colours represent different clusters of voxels that have been recovered.*

5.1.2 Occlusion coverage map

This section describes an extension of the camera coverage maps from Section 2.3. The extension involves the impact of occluders on the camera coverage of the scene. Because occlusions considerably limit the actual camera coverage, this phenomenon should be considered. As an illustration, we show the occlusion coverage map and the camera coverage map next to each other from the indoor tracking in Mol in Figure 5.3.

We can create occlusion coverage maps when the cameras are calibrated, and the locations and shapes of the occluders are known. By incorporating occluders in the coverage map, the number of cameras observing a particular area may decrease. The only way to increase camera coverage in particular areas is by moving cameras to other positions in the scene or increasing the number of cameras. Occlusion inevitably decreases the actual camera coverage. These coverage maps show that occlusion can-



(a) camera coverage map *(b) occlusion coverage map*

Figure 5.3: *Coverage map and occlusion coverage map in the Mol camera setup. The occluders' presence reduces the real camera coverage significantly. Both coverage maps are taken at height $Z = 0$.*

not be ignored and must be handled appropriately. The assumption that a camera can observe everything in the camera coverage map (Figure 5.3a) is not correct.

5.2 SfS with occlusion handling

Depending on the application, occlusion is modelled in different ways. For object detection, for instance, the occurrence of occlusion increases the intra-class variations of an object's appearance significantly. For example, a people detector that wants to cope with occlusion should not only detect a person that is in view entirely but also when a person is only partly visible. Therefore, the person's appearance in the training data of such a detector will show substantial variations in appearance at the risk of increasing the number of false positives. A solution to this problem is to resort to part-based detection to determine the presence of a specific part of an object in the scene rather than the entire object. For instance, Li et al. [54] uses an AND-OR structure to handle occlusion. It organizes object parts into consistently visible parts and optional part clusters, and then represents an object with the consistently visible parts (i.e.,AND) and one of the optional part clusters (i.e., OR).

For applications in 3D reconstruction, occlusion manifests in incomplete silhouettes that originate from one of the methods presented in Chapter 3. In the case of static occluders, the moving object/person cannot be completely observed if they are located behind an occluder from the perspective of a camera, leading to incomplete silhouettes. By combining occlusion information with incomplete silhouettes, objects can still be

Figure 5.4: *Mol dataset: four cameras observe the living area. People walk around in this area. A number of occluders are present such as the coffee table, kitchen table and part of the kitchen cabinets.*

reconstructed completely. The key is to know which areas in an image correspond to occluded region. There are two ways to model occlusion: with occlusion masks or with occlusion depth maps. We will discuss this in more detail when we review different approaches to handle occlusion.

In Chapter 3 we explained the standard shape-from-silhouettes algorithm. This algorithm cannot cope with incomplete silhouettes because the method assumes that all cameras can observe the entire object. This section will explain three approaches from literature that can cope with occlusion to some extent. We compare these methods to our proposed method in the experiments section (see Section 5.4).

To illustrate the different approaches, we consider a dataset recorded in Mol where four cameras observe the living area (Figure 5.4). The coffee table, kitchen table, and part of the kitchen cabinets may cause partial occlusion when people walk around in the scene.

5.2.1 Shape from incomplete silhouettes

Shape from incomplete silhouettes by Snow et al. [91] is a method that relaxes the condition for a voxel to be part of the reconstruction. The method requires the intersection of at least $N - e$ backprojected silhouettes to allow a reconstruction, where e is the number of acceptable misses among the N camera views. If $e \geq 1$, single misses will not remove object parts from the reconstruction in this approach. However, the resulting shape is larger than the visual hull for requiring fewer intersections of visual cones. A drawback of this approach is that larger hulls are reconstructed even if the silhouettes are consistent. Moreover, the parameter e is best kept low to avoid large excesses.

Landabaso et al. provides a similar solution [51]. However, the criteria for classifying voxels as part of the shape happens by minimizing the probability of voxel miss-classification. This method reconstructs the visual hull with standard SfS in the first step. The method reconstructs the

volume with the lowest classification error in silhouette-based systems for those voxels that project consistently to all the camera views. A decision on the voxels not forming part of the visual hull is taken in a second step by minimizing the error probability on each voxel independently. The projection of the computed visual hull is compared to the set of original silhouettes to calculate the error probability.

The biggest drawback of these methods is that they assume that the probability for misses in the silhouettes, either due to erroneous foreground/background segmentation or occlusion, is the same in all images from different cameras. Occluders cause incompleteness in the silhouettes locally, and foreground/background segmentation errors are more likely to happen in some regions of the images due to the similarity of the moving object and the background. Another problem is that these methods do not exploit the fact that occluders typically cause larger areas in a silhouette to be incomplete, which is different from the errors expected from noise in foreground/background segmentation.

5.2.2 SfS using occlusion masks

Guan et al. [37] presented an attempt to handle occlusion in a shape-from-silhouettes-based algorithm by modelling occlusions. Guan used occlusion masks to model pixel-based information per camera. In essence, an occlusion mask is a binary image of the same size as a camera image, indicating the pixels that may be occluded in a view. Figure 5.5 shows the occlusion masks of the dataset recorded in Mol. We see that the couch, kitchen table, coffee table and kitchen cabinet are indicated as occluded regions in the images because a person can walk behind these objects from the camera's perspective. Objects placed against the wall are not part of the occlusion mask because they cannot appear behind them. Hence, they cause no occlusion. Therefore, an occluder may be part of an occlusion mask in one view but not necessarily in another view. Examples of such objects are the kitchen cabinet and one of the couches. We chose to model the occluding regions manually with simple polygonal shapes. The occluders can also be segmented on a pixel level, but that requires more time.

The extension of the standard shape-from-silhouettes algorithms with occlusion masks is relatively straightforward. Guan treats the occluded areas as special areas in the cameras, which means that these regions are not consulted in the reconstruction process. Therefore, the process prevents that the lack of silhouette pixels in these special regions may remove parts of the object.

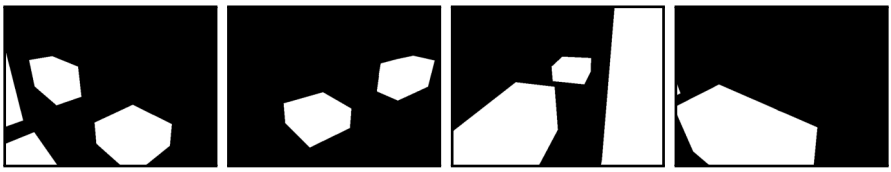


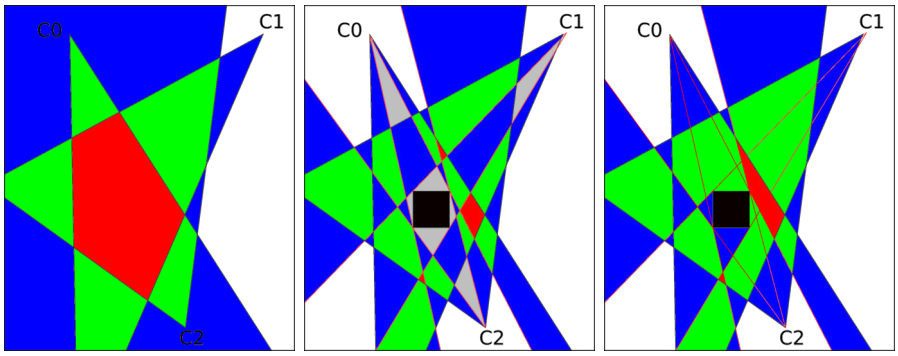
Figure 5.5: *Occlusion masks of the dataset recorded in Mol. White means that a pixel may be occluded, black means that there is no static occluder at that location. For simplicity the occluders were modelled using polygonal shapes.*

Specifically, the evaluation process to accept a voxel v in the shape reconstruction process is slightly adapted. A voxel's projection no longer needs to be part of the silhouette in each view. That is now only the case for the views where the voxel's projection lies outside the special areas. A voxel should at least be part of some silhouettes to be kept in the reconstruction. This rule prevents that the static occluders are reconstructed as well.

The biggest drawback of a binary representation for occlusion is that it cannot handle the difference between the area in front and behind the occluder appropriately. Occluders situated behind an object become less relevant than when this occluder is in front of this object. Silhouettes detected in specials regions of a camera indicate that a person stands in front of the occluder. However, we need to be careful to treat such a case appropriately. If multiple people are in the scene, one in front of the occluder and one behind the occluder, we cannot simply assume that the silhouette inside the special region is complete.

The occlusion masks can be created manually using photo editor software. Alternatively, the occlusion maps could be learned automatically over a period of time, for instance, by using the method presented in [37]. This method introduces the concept of effective boundaries to find occluders in the image. The problem is again that when a person can appear in front and behind an occluder, the occlusion mask will not be generated correctly and the reconstruction based on these erroneous masks will fail.

The main drawback with occlusion masks remains that they do not model depth information appropriately. Depth information is crucial to assess occlusion, and therefore we introduced the use of occlusion depth maps. Moreover, the camera coverage for the reconstruction algorithm is increased because the voxels in front of an occluder will be evaluated as opposed to occlusion masks where the entire line of sight of the special areas decreases the camera coverage.



(a) no occlusion *(b) using masks* *(c) using depth maps*

Figure 5.6: *We show the camera coverage map without occlusion in (a). The camera coverage map is drastically reduced by adding the black occluder in (b) and (c). Occlusion masks (b) artificially reduce the camera coverage even further by not using regions in front of the occluders. The occlusion depth maps in (c) obtain the correct camera coverage.*

5.2.3 SfS using occlusion depth maps

We solve the lack of 3D information in the occlusion masks by introducing occlusion depth maps [85]. In Figure 5.6 we illustrate that the camera coverage significantly increases in multiple areas when using occlusion depth maps instead of occlusion masks.

Like occlusion masks, the dimensions of each occlusion depth map $m_{v,i}$ is equal to the resolution of the observed images of the camera. However, for each pixel, the closest distance is calculated to an occluder along the line of sight. The range is therefore $[0, +\infty[$.

Figure 5.7 shows an example of these depth maps on the dataset recorded in Mol. Unlike the binary decision in the occlusion masks, occlusion depth maps hold information about the precise location of the occluders in the scene and can therefore distinguish between areas in front an behind the occluder. Moreover, it is possible to determine which voxels can be observed by each camera using these occlusion depth maps. Once the calibration is known, it is possible to model occluders in 3D and generate the occlusion depth maps automatically for all views. There is no drawback of modelling occluders, even in views where they will never be occluded, unlike in occlusion masks. What is important, however, is to note that such an occlusion depth map needs to be updated if it also wants to handle dynamic occlusion.

We adapted the standard shape-from-silhouettes concept to include the

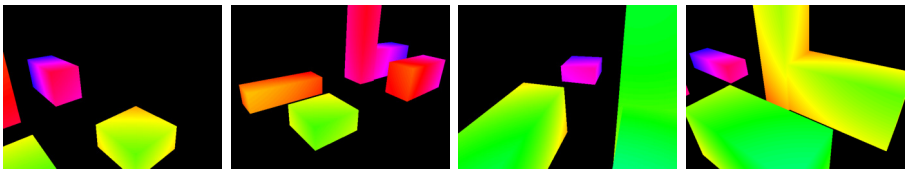


Figure 5.7: *Occlusion depth map of the dataset recorded in Mol. Depth map values are represented from green (close-by) to blue (far away). Black means that no relevant occlusion is present for the reconstruction at hand (e.g., the object of interest will not be occluded by static objects in that part of the scene.*

information of the occlusion depth maps. First, we determined by which cameras each voxel can be observed. Only the part of the silhouettes that corresponds to the observable area of each camera are evaluated. The distance $d_{v,j}$ between each voxel and each camera is calculated. Let $m_{v,j}$ be the corresponding depth map value for the projected voxel v on camera j. If $d_{v,j} \geq m_{v,j}$, the corresponding silhouette of this camera at that position is not used. In contrast, if $d_{v,j} < m_{v,j}$ the silhouette is used, and we check if the voxel's projection is part of the silhouette or not. To reduce computation time, a list of projections was generated per voxel that is used during the reconstruction. Since the occlusion depth maps and camera setup remained static, these projections remained the same as well.

The occlusion depth maps tell which n cameras effectively observe each voxel in the scene, with $0 \leq n \leq N$. The reconstruction process only evaluates these n cameras. Only when the voxel's projections are all consistent, meaning all voxel's projections are part of the silhouette of these cameras, the voxel is part of the reconstruction. Depending on the camera configuration and the number and size of occluders in the scene, n may be small for specific areas in the reconstruction space. To prevent objects from being reconstructed in these areas with camera coverage, we introduce the minimum required camera consistency c_{min}. Only if $n \geq c_{min}$ and the voxel's projection is part of the silhouette in all views, a voxel is part of the reconstruction. We choose a suitable value for the minimum required camera consistency based on the occlusions in the scene, e.g., by using the occlusion coverage map (see Section 5.1.2).

The occlusion depth maps may be generated manually, but we also looked into auto-generating these maps in [84]. The method learns a voxel-based occlusion map, which can be converted to an occlusion depth map. For each voxel in the reconstruction space, the method learns which cameras can observe it. For the method to work, a person needs to walk around in the scene. Figure 5.8 shows a result of this

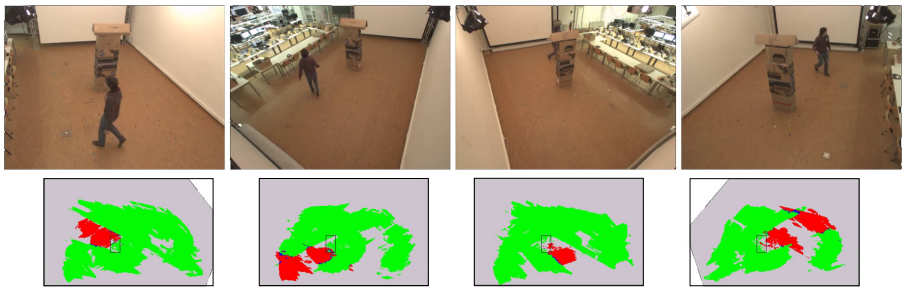


Figure 5.8: *Example of the self-learning map method at Campus Schoonmeersen. A person walks around in the scene and the pile of boxes in the middle of the room cause all cameras to observe an occluded view of the person at some point. After circling twice around the boxes, the occlusion map is built. The occlusion maps are shown at the bottom of the figure. Red areas are occluded for that particular camera while green areas are visible. Gray areas are not yet observed, white areas are out of view.*

self-learning process.

To decide if there is occlusion, we made some assumptions. One of these assumptions is that there needs to be a consensus about the occupation of a voxel in the space. This consensus is only possible if a minimum number of cameras agree that a voxel at a particular position is occupied. Suppose three cameras are agreeing there is an object at a particular position in space, but the fourth camera does not detect anything there. In that case, there might be an occluder positioned between this object and the cameras. However, it is not clear how many cameras need to agree about this visibility. Moreover, the shape-from-silhouettes algorithm carves away parts from the reconstruction space just because a camera does not observe certain voxels. It doesn't necessarily mean that an occluder is present.

Since it is hard to decide whether a voxel is occluded for a specific camera, we count votes for both cases: occluded and non-occluded. If a person's reconstruction occupies a voxel, this voxel is deemed to be observable. In this way, only the truly occluded regions remained. The only requirement for this method to work is that a single person walks around in the scene to make occluded regions observable. This method provided a way of creating occlusion depth maps completely automatically. The biggest drawback is that only static occlusion is modelled and the method remained sensitive to errors in the silhouettes outside the occluded regions of a camera.

5.3 Automatic occlusion detection and handling

The previous section explained how occlusion masks or occlusion depth maps model occlusion. We will now propose an occlusion detection algorithm that automatically determines where the occluders are positioned in the scene and automatically creates the occlusion depth maps purely based on observations. The big advantage of our approach is that such a system can handle both dynamic and static occlusion. This method requires more than four camera views to keep the excess volume limited. In the remainder of this chapter, we explain the different steps of this method and present the results.

At the core of the method is the partitioning of the reconstruction space in cells, i.e. regions with a uniform camera and silhouette coverage properties. These cells are clusters of voxels and have the advantage that reasoning per voxel is not needed, reducing the number of computations considerably. Since all voxels in such a cell have uniform camera and silhouette coverage properties, we determine if an entire cell should be part of the reconstruction or not. An iterative process is proposed, which incrementally adds cells to the temporal reconstruction based on their potential to explain the observed silhouettes from different cameras. Two versions of the algorithm were published: one at the ACIVS 2017 conference in Antwerp [88] and the other as a journal paper in the Integrated Computer-Aided Engineering by IOS Press [89]. The main difference between the two methods is the criterion that decides in what order cells are added to the reconstruction. The first approach uses counting functions to rank the cells, while the second method calculates the actual benefit of adding each of the remaining cells and ranks them accordingly. We will focus on the former method. In section 5.3.4 we will explain the differences with the 2019 publication.

5.3.1 Partitioning of the reconstruction space into cells

Let I_j be the silhouette of an object $\mathcal{S}$ with respect to camera j. We denote the projection of a point $P \in \mathbb{R}^3$ on the image sensor of camera j as P_j. For each point $P \in \mathbb{R}^3$ we define a membership function $\psi_j(P)$ as follows:

$$\psi_j(P) = \begin{cases} 1 & \text{if } P_j \in I_j \\ 0 & \text{otherwise,} \end{cases} \tag{5.1}$$

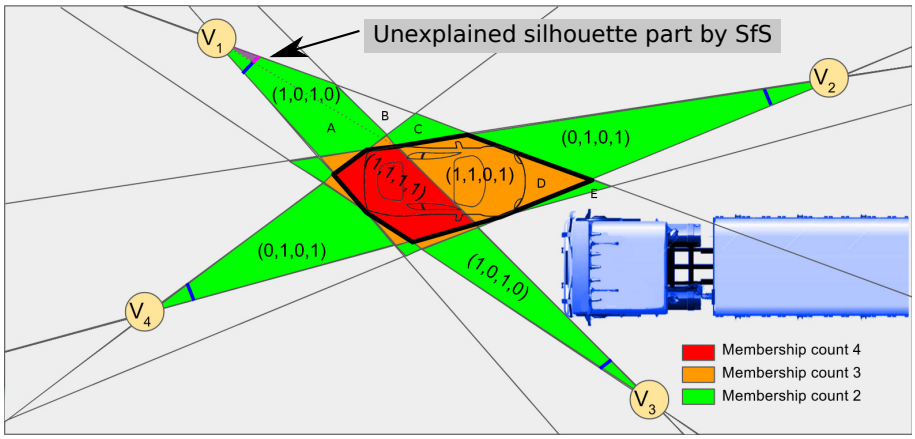


Figure 5.9: *Example of the space partitioning into cells in 2D with four viewpoints in case a stationary truck is partially blocking the view of camera 3. The aim is to find those cells which are part of the car. Different cells with membership count 2, 3 and 4 are coloured as these are the cells that have to be evaluated. In some of the cells, we printed the cell's membership vector $\boldsymbol{\psi}$.*

which indicates whether or not the projection of point P lies inside or outside the silhouette of a particular camera j. For N cameras we define the membership vector

$$\boldsymbol{\psi}(P) = (\psi_1(P), \ldots, \psi_N(P)). \tag{5.2}$$

That is, $\boldsymbol{\psi}(P)$ will be a binary vector of the form $(\ldots, 0, \ldots, 1, \ldots)$ that indicates for which cameras the projection of P lies within the silhouette, and for which cameras it is not. The membership vector $(1, 1, \ldots, 1)$ represents the shape-from-silhouettes of $\mathcal{S}$. The points outside the shape-from-silhouettes are the points P for which at least one element of $\boldsymbol{\psi}(P)$ is zero. When at least one of the silhouettes is incomplete, however, some elements of $\boldsymbol{\psi}(P)$ may be zero even when P belong to $\mathcal{S}$. We will refer to the number of ones in the membership vector as the membership count $\sum_{i=1}^{N} \boldsymbol{\psi_i}$. Figure 5.9 shows some of these membership vectors.

Now let P be any point in $\mathbb{R}^3$, then the cell A is defined as the set of all points $Q \in \mathbb{R}^3$ for which there exists a continuous path from Q to P such that $\boldsymbol{\psi}(P) = \boldsymbol{\psi}(R)$ for all points R along the path. Thus, a cell has the following properties:

1. all points in a cell share the same membership vector;
2. all points in a cell are simply connected;
3. a cell is maximal in the sense that it cannot be a subset of a larger set that has the properties in (1) and (2).

The above definition of a cell can easily be adapted for a 2D scene. Figure 5.9 illustrates the subdivision of 2D space into polygonal cells. In this figure it is assumed that the car is moving and the truck is not. The view of the car is partially blocked by a parked truck. The cells arise from the backprojection of the silhouettes observed by each camera.

The shape-from-silhouettes is equivalent to the union of cells for which the membership count equals N (see Theorem 2). Clearly, the shape-from-silhouettes is a poor approximation of the visual hull when one or more silhouettes are incomplete. Let us illustrate this with the example in Figure 5.9. The tick lines in front of each camera represent the silhouettes. Cameras V_1, V_2 and V_4 can observe the entire car. The truck occludes the front of the car in camera V_3. The corresponding shape-from-silhouettes reconstruction is the red area of the car having a membership count of 4. This reconstruction is incomplete because the front of the car is missing. To completely reconstruct the car, cell D should also be part of the reconstructed shape. Cell D is seen from cameras V_1, V_2 and V_4, but not from V_3. The interesting part is that camera 1 will agree about this cell because it would explain the cyan part of its silhouette. The way the algorithm will work is to add cells that explain missing silhouette parts. The union of shape-from-silhouettes and cell D is a shape that almost equals to the visual hull as if the occluder had not been present.

The approximated shape that we want to find is the minimal union of all the cells A^k, that contain at least one point of the visual hull, where K is the set all cell indices k in the reconstruction space. We can write this as:

$$\bigcup_{k \in K} A^k, \text{where} A^k \cap \boldsymbol{VH}(\mathcal{S}, \mathcal{V}) \neq \emptyset. \tag{5.3}$$

Note however that formula (5.3) does not provide a method for finding the cells A^k, since we do not know $\boldsymbol{VH}(\mathcal{S}, \mathcal{V})$. Hence, we need other criteria to decide which A^k are part of this reconstructed shape.

5.3.2 Cell-based geometric reasoning

We know that the result should be a combination of cell $A^k k$. The first difficulty, however, is that it is not feasible to test all possible cell combinations. Since the number of these combinations scales exponentially in the number of cells (which easily exceeds 60 in a real-world experiment with eight cameras), a strategy is needed to navigate the

search space quickly. Unfortunately, it is not possible to find the correct cell combination from a single measurement. Therefore, we propose an iterative approach that converges toward the correct shape.

A second challenge is that parts of a silhouette can be explained by multiple cells. Reconsidering Figure 5.9, the missing cyan part in the silhouette of camera V_1 (given the shape-from-silhouettes reconstruction, the red cell), can be explained by cells A, B, C, D and E. It also illustrates that once a cell is added to the reconstruction (e.g., cell D), other cells may become superfluous.

The iteration starts from a reconstruction that equals the shape-from-silhouettes $\boldsymbol{SfS}(\mathrm{I}, \mathcal{V})$. This as an logical starting point since this cell is view-consistent: the projection of this cell is part of the silhouette in each camera and, hence, part of the reconstruction. This initial shape Y^0 is projected onto each of the camera views. We define define $I(Y^0)_j$ as the projection of cell Y^0 on camera j. This projected shape is part of the silhouette, so $I(Y^0)_j \in I_j$. Some or all of the silhouettes may not yet be covered by $I(Y^0)_j$. The goal is to find a set of cells that explain the remaining part of these silhouettes in a way that is as consistent as possible in all camera views.

At each iteration a cell will be added. The iteration thus proceeds as follows:

$$Y^t = Y^{t-1} \cup A^{k_m}, \tag{5.4}$$

where A^{k_m} indicates the cell which corresponds to the highest consistency score at iteration $t-1$. This consistency score is a measure to indicates by how much the consistency improves in the silhouettes by adding this cell. The consistency is improved when a cell projects inside unexplained regions of silhouettes that have not yet been explained by cells in the reconstruction. Consistency may also deteriorate by adding unnecessary parts to some silhouettes. The consistency score will be explained in the following section.

I. Evaluation metric

We will explain the choice for our consistency score by first looking at the end goal. A method such as the one presented in this work needs an objective evaluation to compare different solutions. Two shapes can be compared by analysing their voxels. One of the shapes is used as the reference shape, the other shape is compared with the reference shape. Each voxel in the reconstruction space is classified in one of four classes: TP (true positive), FP (false positive), FN (false negative) and

TN (true negative). From these values, precision and recall is calculated. This concept has previously been used in Section 4.5.2. The evaluation of how well a shape resembles the reference shape is measured using the F_β-score.

$$F_\beta = (1+\beta^2)\frac{\text{precision . recall}}{(\beta^2\text{precision}) + \text{recall}}. \tag{5.5}$$

The parameter β can be chosen depending on the application and balances the weights between precision or recall. The F_1 score is the harmonic mean of the precision and recall.

While we want to approximate the 3D shape of the object as closely as possible as in Equation 5.3, in a real application, only the silhouettes are available. Therefore, we do not compare reconstructions directly, but their projections in the different views. Instead of comparing voxels, we compare pixels and we aim to maximize the F-score. The reconstruction method adds cells one by one such that the F-score in (5.5) increases by an amount ΔF at each iteration. However, since ground truth is not available (the silhouettes without occlusion), the exact ΔF cannot be calculated precisely. Therefore, we propose a technique that estimates ΔF on extended silhouettes, which proves to work well.

II. Extended silhouettes

As our consistency score needs to reflect whether a cell explains part of a silhouette that was not yet explained by earlier additions, the silhouette we compare to should be a combination of the observed silhouette I_j, and the projection of the reconstructed volume at iteration t, $I(Y^t)_j$, rather than only the observed silhouette, because we know that the incomplete silhouette are not a correct estimation of the silhouettes without occlusion. We represent the extended silhouette at iteration t of camera j with the newest addition of cell A^{k_m} as follows:

$$I^t_{E,j} = I^{t-1}_{E,j} \cup I(A^{k_m})_j, \tag{5.6}$$

where $I^0_{E,j} = I_j$, the observed silhouettes and $I(A^{k_m})_j$ represents the projection of cell A^{k_m} on camera j.

The consistency scores for the remaining cells in the search space need to be recalculated each time a cell is accepted because the extended silhouettes change. In fact, as the reconstruction Y^t grows at every iteration, the extended silhouettes $I^t_{E,j}$ grow accordingly, and, therefore,

the consistency scores of the remaining cells may change. The iterative adaptation of the silhouettes allows to find a maximum consistency score and a clear stop criterion because the F-score will not increase any further after some iterations.

III. Coverage, resemblance and consistency score

We will compare how well a cell fits the extended silhouettes using scores inspired by recall and precision as in (5.5), as these describe the properties of overlap and excess well. Since the comparison is not against the ground truth, however, we will not use the terms recall and precision not to confuse the reader. Instead, we will define coverage and resemblance as the equivalent comparison metrics against extended silhouettes.

From (5.6) we know that the extended silhouettes depend on the previous iteration step $t-1$. Coverage and resemblance in this section are calculated for candidate extensions of the current reconstruction. A candidate extension is denoted as $H_{t,k} = Y^{t-1} \cup A^k$.

Coverage on camera j is defined as the fraction of the extended silhouette that is covered by the projection of $H_{t,k}$ denoted as $I(H_{t,k})_j$ (Figure 5.10a):

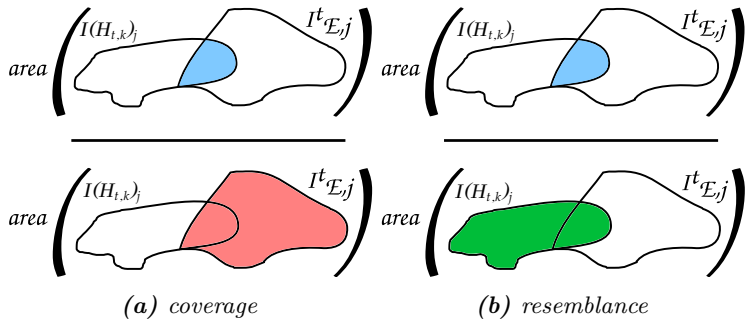
$$\mathrm{cov}_j(I^t_{E,j}, I(H_{t,k})_j) = \frac{\mathrm{area}(I(H_{t,k})_j \cap I^t_{E,j})}{\mathrm{area}(I^t_{E,j})}. \tag{5.7}$$

Resemblance on camera j is defined by the fraction between the area of the projected shape $I(H_{t,k})_j$ that is part of the extended silhouette $I^t_{E,j}$, and the area of the projected shape $I(H_{t,k})_j$ (Figure 5.10b):

$$\mathrm{res}_j(I^t_{E,j}, I(H_{t,k})_j) = \frac{\mathrm{area}(I(H_{t,k})_j \cap I^t_{E,j})}{\mathrm{area}(I(H_{t,k})_j)}. \tag{5.8}$$

Figure 5.11 shows an example of the iterative progress of coverage and resemblance for an occluded view and averaged over all camera views. Even when the resemblance drops in iteration 4 for the occluded camera, we see that the average resemblance still increases.

Based on Equation 5.5, we estimate an F-score per camera, where precision is estimated by resemblance and recall by coverage. Equations



(a) *coverage* **(b)** *resemblance*

Figure 5.10: *Coverage (a) and resemblance (b) based on the projection of a shape $I(H_{t,k})_j$ and the extended silhouette $I^t_{E,j}$ (equivalent of ground truth). Each image represents the division of the area of the coloured areas, equivalent to precision and recall.*

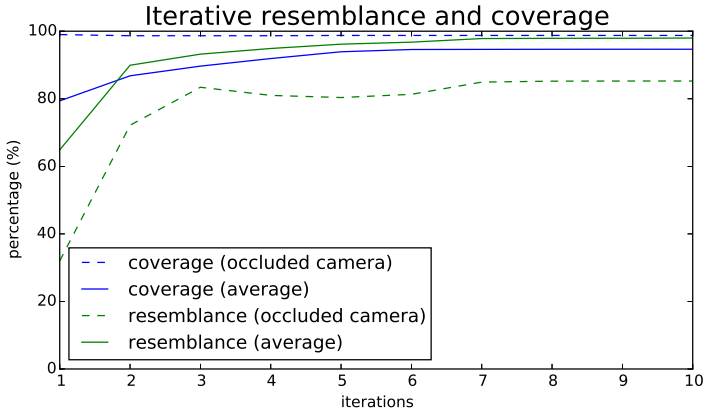


Figure 5.11: *Typical iterative coverage and resemblance graph of an occluded camera view and the averaged coverage and resemblance over all views.*

5.7 and 5.8 are used to determine the value $\lambda_{\beta,j}(I^t_{E,j}, I(H_{t,k})_j)$ for each view:

$$\lambda_{\beta,j} = (1+\beta^2)\frac{\text{res}_j \ . \ \text{cov}_j}{(\beta^2 \text{res}_j) + \text{cov}_j}, \tag{5.9}$$

where we omit the parameters $I^t_{E,j}$ and $I(H_{t,k})_j$ for simplicity. The value of β will be discussed in the next part.

For a scenario where N cameras are equally important to the reconstruction, we consider the estimated F-scores of the cell regarding the global reconstruction as the average of all $\lambda_{\beta,j}$-scores. The consistency score

$\Lambda_\beta(I_E^t, H_{t,k})$ of candidate solution $H_{t,k}$ is then

$$\Lambda_\beta(I_E^t, H_{t,k}) = \frac{1}{N}\sum_{j=1}^{N} \lambda_{\beta,j}(I_{E,j}^t, I(H_{t,k})_j). \tag{5.10}$$

Furthermore, we assume that the F-score of the 3D reconstruction as in (5.5) is related to $\Lambda_\beta(I_E^t, H_{t,k})$, where the latter is computed by taking the average of estimated F-scores based on 2D silhouettes. We assume that any increase of $\Lambda_\beta(I_E^t, H_{t,k})$ is expected to increase F.

This consistency score is calculated at each iteration step for each cell A^k in the search space by testing the corresponding candidate solution $H_{t,k}$. If the consistency score $\Lambda_\beta(I_E^t, H_{t,k})$ is larger than the current consistency score $\Lambda_\beta(I_E^t, Y^{t-1})$, the cell is kept in the search space. If not, the cell is removed from the search space. This mechanism reduces the search space at each iteration step.

The cell A^{k_m} with the highest consistency score is added to the current reconstruction $Y^t = Y^{t-1} \cup A^{k_m}$ and also removed from the search space.

The stop criterion in this optimization process is straightforward. Once no cell in the search space can improve the consistency score, the algorithm terminates. Since the number of cells is finite, the algorithm will always stop.

IV. Coverage vs. resemblance

Depending on the application, either coverage or resemblance may be more important. For example, the reconstruction of a person working closely together with a robot, should reconstruct all parts of the person to ensure safety margins between the person and the robot. Therefore the coverage is most important. In cluttered real-world scenes with many spurious foreground detections, resemblance becomes more important to avoid that cells are included in the reconstruction, which are barely part of the object that is being reconstructed. The value of β controls the balance between coverage and resemblance.

Typical values for β are between 0.5 and 10. If a system needs to be calibrated, we propose to start with β equal to 1 and adapt β for a typical scene until the subjective assessment of an operator in terms of connectedness, convexity or any other measure is met. The optimal value of β also depends on the quality of the camera calibration and silhouette detector. A sensitivity analysis of parameter β is provided in Section 5.4.2.

Objective
Given N calibrated observed (incomplete) silhouettes. Find the subset of cells that are part of the visual hull.

Algorithm
(i) **Divide the reconstruction space into cells (Section 5.3.1)**

(ii) **Compute the reconstructed shape:**
(a) **Find all cells** that project **inside each silhouette** of its observing cameras:
$Y^0 = \{P \in \mathbb{R} : \psi_j(P) = 1, \forall j\}$
$K^0 = \{k : A^k \not\subset Y^0\}$
(b) **Set** $I^0_{E,j} = I_j$ for all camera views j
(c) **Iterative reconstruction (iteration t while $K^t \neq \emptyset$):**

$k_m = \underset{k \in K^t}{\operatorname{argmax}}\ \Lambda_\beta(I_E^{t-1}, Y^t \cup A^k)$
If $\Lambda_\beta(I_E^{t-1}, Y^{t-1} \cup A^{k_m}) > \Lambda_\beta(I_E^{t-1}, Y^{t-1})$ **:**
$Y^t = Y^{t-1} \cup A^{k_m}$
$I^t_{E,j} = I^{t-1}_{E,j} \cup I(A^{k_m})_j$
$K^t = K^{t-1} \setminus \{k_m\}$
Else:
stop
return Y^{t-1}

Algorithm 5.1: *The proposed shape reconstruction method from incomplete silhouettes. We have omitted the pruning of unfit cells from the above algorithm for simplicity. Note that the algorithm terminates when there are no incomplete silhouettes.*

V. The occlusion handling algorithm

Algorithm 5.1 shows the different steps of the proposed method as explained in the previous sections. In a practical implementation, we often reconstruct an object as a set of voxels, and we define a cell as a set of voxels. However, the algorithm can also be implemented without discretization to a voxel space.

Cameras have limited fields of view. Therefore parts of the scene may lay outside of the FOV of a camera. Equations 5.7 and 5.8 only consider the part of the scene which project inside the FOV of each camera. In Section 5.4 some experiments are conducted where the object of interest

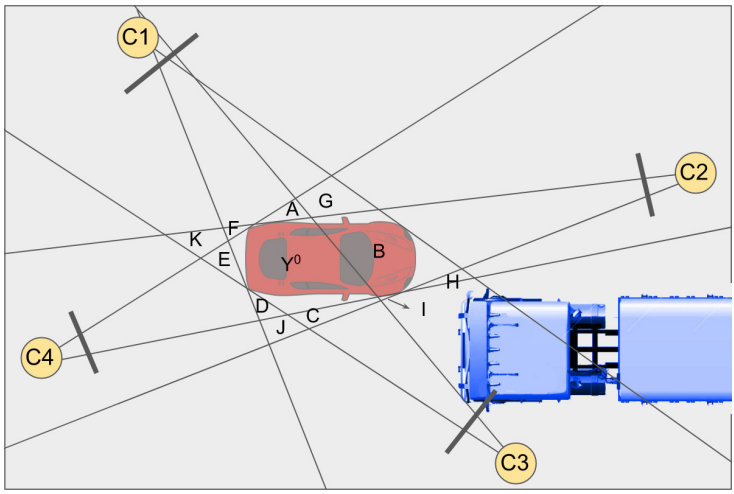


Figure 5.12: *Example with one stationary truck, acting as a static occluder for cameras* 3*.*

is not inside the FOV of all cameras to illustrate this feature. Also the initial reconstruction is different from the strict definition of $\boldsymbol{SfS}(\mathrm{I}, \mathcal{V})$. The initial shape becomes the union of all cells which project inside the observed silhouettes, only considering the cameras for which the cell projects inside the camera's FOV.

In Equation 5.10, we assumed that all cameras are equally important. In practical applications, camera calibration and silhouette extraction using foreground/background segmentation are subject to inaccuracies and noise. Cameras closer to the object of interest can usually observe the object more accurately and the calibration of the camera may be more reliable than cameras calibrated from a distance. Therefore, as an extension, we propose the use of weights $w_{j,k}$ per camera j and per cell A^k depending on the average distance between the camera and the cell. Let $d_{j,k}$ be the average distance between the cell A^k and camera j. Then we can normalize the weights as follows:

$$w_{j,k} = \frac{d_{j,k}}{\sum_{j=1}^{N} d_{j,k}}. \tag{5.11}$$

Therefore, we can rewrite Equation 5.10 as:

$$\Lambda_\beta(I_E^t, H_{t,k}) = \sum_{j=1}^{N} w_{j,k} \lambda_{\beta,j}(I_{E,j}^t, I(H_{t,k})_j). \tag{5.12}$$

it	cell	mc	res_1	res_2	res_3	res_4	cov_1	cov_2	cov_3	cov_4	Λ_β	?
0	Y^0	4	1.00	1.00	1.00	1.00	.60	.80	1.00	1.00	.871	A
1	$\underline{Y^0}$	4	1.00	1.00	1.00	1.00	.60	.80	1.00	1.00	.871	-
	A	3	1.00	.67	1.00	1.00	.61	.80	1.00	1.00	.858	R
	B	3	1.00	1.00	.31	1.00	1.00	1.00	1.00	1.00	**.922**	A
	C	3	1.00	1.00	1.00	.83	.60	1.00	1.00	1.00	.903	K
	D	3	1.00	1.00	.83	1.00	.60	.80	1.00	1.00	.862	R
	E	3	.71	1.00	1.00	1.00	.60	.80	1.00	1.00	.863	R
	F	3	1.00	1.00	1.00	.83	.60	.80	1.00	1.00	.862	R
	G	2	1.00	.67	.50	1.00	1.00	.80	1.00	1.00	.901	K
	H	2	1.00	.80	.40	1.00	.90	.80	1.00	1.00	.872	K
	I	2	1.00	1.00	.73	.90	.70	1.00	1.00	1.00	.913	K
	J	2	1.00	1.00	.67	.83	.60	1.00	1.00	1.00	.881	K
	K	2	.63	1.00	1.00	.63	.60	.80	1.00	1.00	.833	R
2a	$\underline{B}$	3	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.000	-
	C	3	1.00	1.00	1.00	.83	1.00	1.00	1.00	1.00	.990	R
	G	2	1.00	.75	.50	1.00	1.00	1.00	1.00	1.00	.943	R
	H	2	1.00	.84	.62	1.00	1.00	1.00	1.00	1.00	.963	R
	I	2	1.00	1.00	1.00	.90	1.00	1.00	1.00	1.00	.995	R
	J	2	1.00	1.00	.87	.83	1.00	1.00	1.00	1.00	.983	R
2b	$\underline{B}$	3	1.00	1.00	.31	1.00	1.00	1.00	1.00	1.00	.922	-
	C	3	1.00	1.00	1.00	.83	1.00	1.00	1.00	1.00	.990	?
	G	2	1.00	.67	.50	1.00	1.00	1.00	1.00	1.00	.936	?
	H	2	1.00	.80	.31	1.00	1.00	1.00	1.00	1.00	.911	?
	I	2	1.00	1.00	1.00	.90	1.00	1.00	1.00	1.00	.995	?
	J	2	1.00	1.00	.27	.83	1.00	1.00	1.00	1.00	.902	?

Table 5.1: *Example of the algorithm with a stationary truck as occluders for camera 3, captured by four cameras and $\beta = 2$. At iteration 1 we list all candidate cells, which form the search space. The last column indicates the cell's decision: keep (K), add (A) and reject (R). A cell is rejected if the consistency score is lower than the previously accepted cell in the updated reconstruction. The algorithm adds the cell with the highest consistency score at each iteration as long as the consistency score increases. Iteration 2a shows the final iteration of the algorithm when extended silhouettes are used. Iteration 2b is the second iteration in case the observed silhouettes are used. Note that the stop criterion is unclear in the latter case.*

5.3.3 Examples of how the method works

Figure 5.12 shows an example of four cameras observing a red car. A parked truck is partially occluding the car in camera view C_3. We illustrate how the proposed consistency score Λ_β is used to the car entirely despite the occluding view by listing all values in Table 5.1. The algorithm initializes with cell Y^0, which corresponds to a consistency score of 0.871. At the first iteration, the resemblance and coverage are calculated based on the extended silhouettes $I_j \cup I(Y^0)_j$ for each camera j, which is basically I_j in the first iteration. The last column indicates the cell's decision: keep (K), add (A) or reject (R). In the next iteration, cells A, D, E, F and K are already rejected because

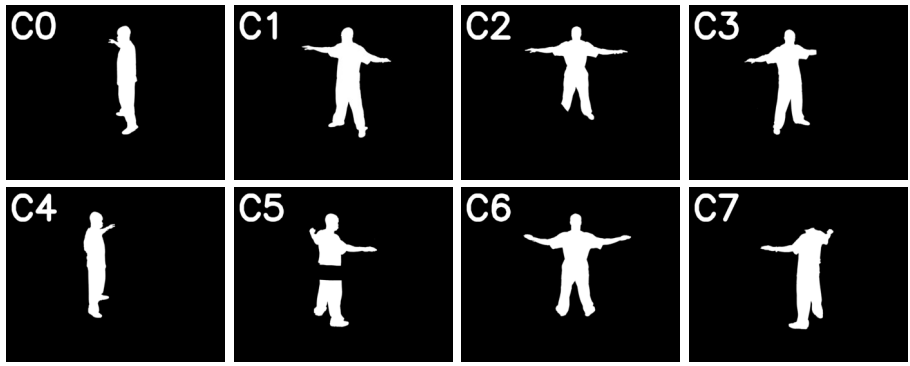


Figure 5.13: *The observed silhouettes from eight cameras are shown. The silhouettes of C_2, C_5 and C_7 are incomplete due to occlusion. The left foot, the waist and the head are missing from the silhouettes in these occluded views.*

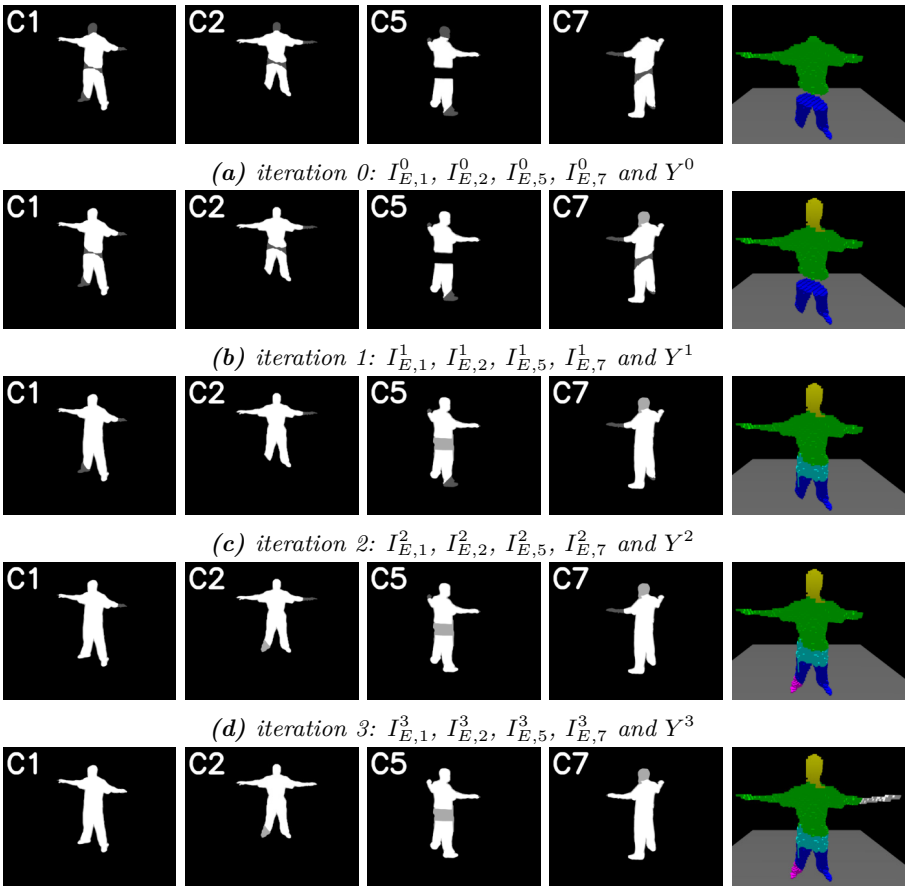
their consistency score is lower than the current consistency score. The highest consistency score, 0.922, corresponds to cell B. This cell is added to the reconstruction, which is now $Y^1 = Y^0 \cup B$.

Possible candidates in iteration 2 are cells C, G, H, I and J. Notice that in the table, we have iteration steps 2a and 2b. 2a uses the extended silhouettes, whereas 2b uses the observed silhouettes (just for illustration). In 2a we clearly see that after updating the extended silhouettes, we cannot find any cell that corresponds to a higher consistency score than $Y^0 \cup B$. However, in the case of iteration 2b it is unclear when to stop because the remaining candidate cells C, G, H, I and J produce a higher consistency score.

This example showed that our method works in 2D. The concepts are the same in 3D but the silhouettes are no longer 1D line segments, but 2D images. Figure 5.13 shows the input silhouettes of a 3D example. We use eight camera views, in which the person is partly occluded in the silhouettes of C_2, C_5 and C_7. The aim is to reconstruct the person without missing parts.

Figure 5.14 shows how the method recovers missing parts in the silhouettes in each iteration. At iteration 0, Y^0 is reconstructed. Clearly, the missing parts in the silhouettes are not reconstructed. We also see that this reconstruction consists of two cells because they are not connected due to the silhouette in C_5.

For each iteration, we show four extended silhouettes, one view C_1 with complete silhouette and the three occluded views C_2, C_5 and C_7. The extended silhouette corresponds, in fact, to the non-zero pixels in each shown silhouette. However, to visualize what is happening we introduce



(a) iteration 0: $I^0_{E,1}$, $I^0_{E,2}$, $I^0_{E,5}$, $I^0_{E,7}$ *and* Y^0

(b) iteration 1: $I^1_{E,1}$, $I^1_{E,2}$, $I^1_{E,5}$, $I^1_{E,7}$ *and* Y^1

(c) iteration 2: $I^2_{E,1}$, $I^2_{E,2}$, $I^2_{E,5}$, $I^2_{E,7}$ *and* Y^2

(d) iteration 3: $I^3_{E,1}$, $I^3_{E,2}$, $I^3_{E,5}$, $I^3_{E,7}$ *and* Y^3

(e) iteration 4: $I^4_{E,1}$, $I^4_{E,2}$, $I^4_{E,5}$, $I^4_{E,7}$ *and* Y^4

Figure 5.14: *Proposed 3D reconstruction with occlusion handling. For each iteration (a) to (e), the extended silhouettes of* C_1, C_2, C_5 *and* C_7 *are shown. The extended silhouettes* I^t_E *evolve into complete silhouettes. Legend: dark grey:* $I^t_{E,j} \setminus Y^t_j$, *light grey:* $I^t_{E,j} \setminus I_j$, *white:* $I_j \cap I^t_{E,j}$.

different shades. Light grey regions correspond to regions that represent recovered parts (previously incomplete parts of the silhouette), these regions correspond to $I^t_{E,j} \setminus I_j$. Dark grey regions correspond to regions $I^t_{E,j} \setminus Y^t_j$, which means that they are not yet part of the reconstruction, but are part of the extended silhouettes (and also of I_j). The white regions correspond to the set of pixels where $I_j \cap I^t_{E,j}$.

At each iteration a part of the person is added to the reconstruction Y_t, more specifically, the head, the waist, the right foot and the left arm in iteration 1, 2, 3 and 4, respectively. In the last iteration we see that the silhouettes now look as if they where taken from unoccluded views.

Type	Description
I	unnecessary, but unharmful (part of input silhouette)
II	unnecessary, but unharmful (part of detected occluded area)
III	adding cell better explains I_j
IV	unnecessary, but may be part of previously detected occluder
V	unnecessary, and creates additional occluder

Table 5.2: *Cell types and their meaning. Note that the cell type is specific to the considered camera.*

5.3.4 Cell-based approach for 3D reconstruction from incomplete silhouettes (ACIVS 2017)

Before we continue to analyse the results, we want to discuss an alternative method [88] that we developed before the publication of the main method that was just explained. The difference in this method is the mechanism that chooses the most fitting cell at each iteration. Unlike calculating the estimated benefit of adding a particular cell to the existing reconstruction, each cell is classified for each view in one of the types presented in Table 5.2.

A cell's type may be different for each camera view. For example, cell A^k may be of type I for camera 1, but of type III for camera 2. Whether a cell will be added to the reconstruction should depend on how the cell is observed by each camera. To arrive at a ranking system, we introduce type counting functions $\chi_X(A^k, Y^t)$, one for each type, that count how many of the N cameras classify the cell as being of type X. The counting function not only depends on the cell A^k, but also on the current shape approximation Y^t, since Y^t will change during the reconstruction. For example, in a configuration of 4 cameras, if cell A^k is classified as being of type III, III, I, II, respectively in camera views 1 to 4, then $\chi_{\mathrm{I}}(A^k, Y^t) = 1$, $\chi_{\mathrm{II}}(A^k, Y^t) = 1$, $\chi_{\mathrm{III}}(A^k, Y^t) = 2$, $\chi_{\mathrm{IV}}(A^k, Y^t) = 0$, $\chi_{\mathrm{V}}(A^k, Y^t) = 0$.

The same iterative process is used compared to the 2019 version. Only, a simple mechanism is used to select the most appropriate cell A^k. This selection is based on the current values of the counting functions. More

Data: Original silhouettes I_j
Result: reconstructed object (Y)
Set Y^0 equal to the union of cells with membership count N, $t = 0$
Use the silhouettes to construct the list of all cells Φ that are not in Y^0
while $\chi_{\text{III}}(A^k, Y^t) > 0$ *for at least one cell in the list* Φ **do**
  compute the score vectors $\mathrm{W}(A^k, Y^t)$
  sort the cells using a descending lexicographic order on $\mathrm{W}(A, Y^t)$
  select the cell A^k that comes first in the sorted list
  set $Y^{t+1} = Y^t \cup A^k$, remove A^k from the list Φ and increment t by one
end

Algorithm 5.2: *Occlusion detection and handling algorithm. At each iteration, all cells are ranked in descending lexicographic order of there scoring vectors until there are no type* III *cells left.*

precisely, for each cell we define the *score vector*

$$\mathrm{W}(A^k, Y^t) = \Big(\chi_{\text{III}}(A^k, Y^t), \chi_{\text{I}}(A^k, Y^t) + \chi_{\text{II}}(A^k, Y^t), \chi_{\text{IV}}(A^k, Y^t)\Big), \tag{5.13}$$

whose elements are simple linear combinations of the counting functions.

After computing the vector $\mathrm{W}(A^k, Y^t)$ for all cells that are not part of Y^t, the cells are sorted using a descending lexicographic order on their score vectors $\mathrm{W}(A^k, Y^t)$. In front of the list will be the cells that have the largest value for $\chi_{\text{III}}(A^k, Y^t)$. These are cells that match the silhouette I_j of many cameras, but are not yet part of the current reconstruction Y^t. When two cells have the same value for $\chi_{\text{III}}(A^k, Y^t)$ a further distinction is made based on the value of $\chi_{\text{I}}(A^k, Y^t) + \chi_{\text{II}}(A^k, Y^t)$. Thus, cells whose projection is already covered by the projection of Y^t for one or more camera views, will have a larger priority in the list. Finally, when there is still a draw between cells, $\chi_{\text{IV}}(A^k, Y^t)$ is used in order to give preference to cells whose projection is at least partially covered by $I(Y^t)_j$ in one of the cameras.

The reconstruction is now straightforward. After sorting the cells, the cell A^k that is in front of the list is added to Y^t, and the score vectors $\mathrm{W}(A^k, Y^{t+1})$ are recomputed for all cells that are not in Y^{t+1}. This process is repeated as long as there are cells for which $\chi_{\text{III}}(A^k, Y^t) > 0$. Algorithm 5.2 shows the pseudocode of this algorithm.

This approach performs slightly worse than our main approach that uses coverage and resemblances measures because the actual benefit of adding

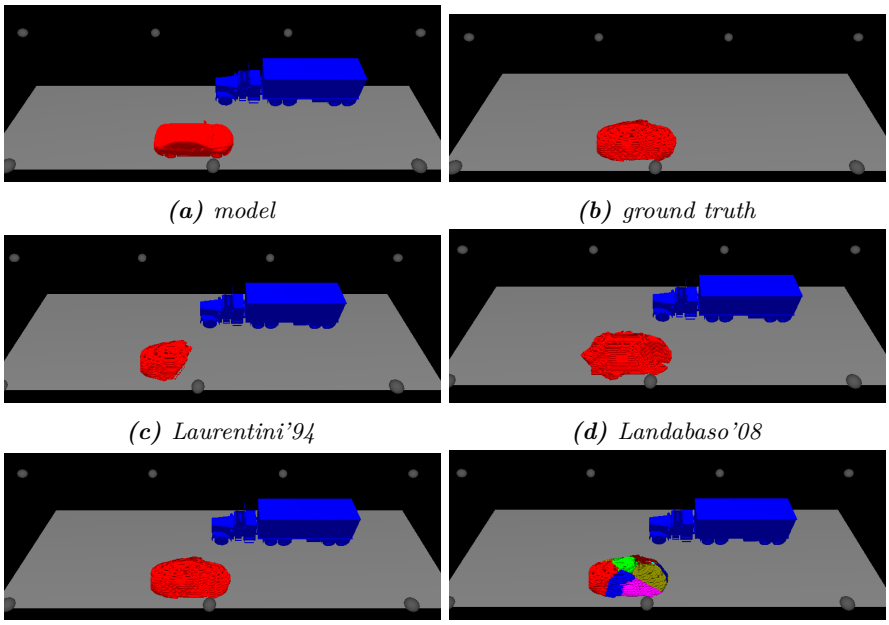
Method	Description
Laurentini'94 [52]	Standard shape-from-silhouettes: Shape-from-silhouettes algorithm, not taking into account the possibility of occlusion.
Guan'06 [37]	Occlusion mask reconstruction: OR-operation between occlusion mask and silhouettes.
Landabaso'08 [51]	Shape from inconsistent silhouettes: Shape-from-silhouettes which keeps all voxels projecting within the silhouettes of at least $N - e$ views, e = number of occluders

Table 5.3: *Description of the comparison methods.*

a cell to the reconstruction is not taken into account here. However, the results of this method are still better than other approaches in literature.

5.4 Experiments and results

We present three experiments that were conducted to show the potential of our method. We compare against four classes of methods from literature that use different approaches (Table 5.3). We simulated *Guan'06* [37] by using perfect occlusion masks because we lack the source code of the actual method. The real performance of this method will most likely be worse than the reported results because the occlusion mask that are automatically generated will not be perfect. Also, *Guan'06*, learns a static occlusion mask over time and, therefore, fails in the case that a moving object can both appear in front as well as behind a static object because depth is not taken into account. *Landabaso'08* [51] represents the set of methods where a parameter e is available and allows that $N-e$ cameras may disagree that a part is reconstructed. This parameter needs to be chosen carefully, depending on the actual camera coverage in the scene. We report two results for this method, one where e equals the number of occluders in the scene to allow a full reconstruction and one extra camera to increase precision at the cost of full reconstruction because the camera coverage is not the same everywhere in the scene. The approach explained in the previous section, *Slembrouck'17* [88] will also be used to compare when relevant. The reconstructed shapes are evaluated at the level of voxels against the ground truth, which is the shape-from-silhouettes where the occluders have been removed. For that reason we resorted to simulation to evaluate the methods in the first and second experiment. In the third experiment we analyse results differently.



(a) model *(b) ground truth*

(c) Laurentini'94 *(d) Landabaso'08*

(e) Guan'06 *(f) proposed*

Figure 5.15: *The simulation consists of a car moving from left to right, captured by 7 cameras. The blue truck on the opposite side of the road is static and occludes the car in the 3 upper most cameras on the right, depending on the location of the car. Different colours in the proposed algorithm indicate different cells.*

5.4.1 Experiment 1: smart traffic analysis

The first experiment concerns smart traffic. We must, however, recall the outdoor experiments from Chapter 3 based on the traffic intersection dataset in Ghent, where we saw that the real-time silhouette extraction methods performed poorly. However, an offline analysis with a precise 3D reconstruction of the scene still has its merits, e.g., for a detailed risk analysis of the traffic at an intersection.

This experiment aims to make detailed reconstructions of vehicles in traffic situations to generate reliable shapes of vehicles on the road. The silhouettes of moving cars are obtained by projecting the 3D models on the different virtual camera sensors and taking into account occluded parts. In the real world, this segmentation can be obtained through FGBG segmentation or using neural network segmentation, such as YOLACT (see Chapter 3). The cameras are mounted on street and traffic lights to create a realistic setting.

method	**precision**	**recall**	**F1-score**
Automatic methods			
Laurentini'94	1.000	.396	.567
Landabaso'08 ($e = 2$)	.817	.904	.859
Landabaso'08 ($e = 3$)	.561	1.000	.718
proposed	.947	.922	.934
Semi-supervised method			
Guan'06	.945	1.000	.972

Table 5.4: *Results for a car going straight, occluded by a parked truck. The proposed method performs almost as well as the supervised method of Guan'06 in terms of precision, recall and F1-score. Other methods are less accurate. We show the results of Landabaso with both $e = 2$ and $e = 3$ to show the 100% recall when e equals the number of occluders.*

Figure 5.15 shows the first situation. A stationary truck is blocking the view on a red car in multiple camera views. The red car at the bottom drives straight from left to right. *Laurentini'94* is unable to reconstruct the complete car (in the example, only 30% of the car is reconstructed). When the car is completely next to the truck, because of occlusion, not a single voxel is reconstructed. *Landabaso'08* performs better because more voxels are found. However, that result includes many voxels that are not part of the car. Our cell-based approach achieves scores comparable to *Guan'06* without using prior knowledge about the scene.

Table 5.4 shows that *Guan'06* achieves the highest $F1$-score followed by our proposed algorithm. The precision of *Laurentini'94* is almost 100%, but its recall is considerably less. Whereas with *Landabaso'08*, the recall is above 90%, but the precision is lower. In Figure 5.15, we see that the proposed reconstruction is much more accurate than the reconstructions produced by the *Landabaso'08* and *Laurentini'94* and comparable to *Guan'06*. The ground truth in this context is the visual hull of the model in Figure 5.15a when the truck is not present. However, it is important to note that the method of *Guan'06* requires external knowledge about the occluders in the scene. The results of *Slembrouck'17* are not available in this experiment because that version cannot handle the limited camera coverage correctly.

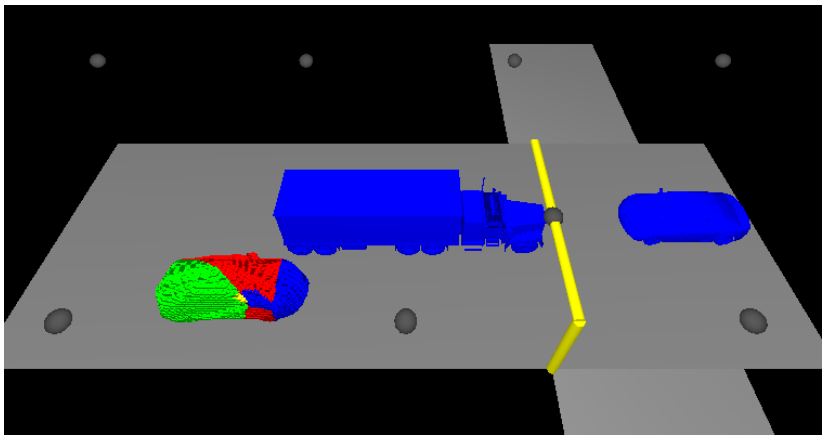


Figure 5.16: *Typical traffic situation where both the blue truck and the blue car want to turn to their left. The car is unable to see the oncoming traffic due to the truck. One extra camera is mounted on the traffic lights (yellow pole).*

method	**precision**	**recall**	**F1-score**
Automatic methods			
Laurentini'94	1.000	.093	.171
Landabaso'08 ($e = 2$)	.744	.837	.788
Landabaso'08 ($e = 3$)	.486	1.000	.654
proposed	.900	.952	.925
Semi-supervised method			
Guan'06	.918	.998	.957

Table 5.5: *Results for an oncoming car with a truck in the middle of the road in front of a car, trying to cross the street in terms of precision, recall and F1-score. Again, the results of the proposed method are close to the semi-supervised method of Guan'06.*

The second situation is a common one in traffic (Figure 5.16). The blue car wants to turn left at the intersection. The truck also wants to turn left from the opposite direction. Since the truck is blocking the view of the oncoming traffic from the blue car's perspective, the blue car has to wait to cross safely. To ensure a safe crossings, we use a multi-camera network to reconstruct the vehicles in the scene.

The proposed algorithm detects oncoming traffic and can evaluate if the crossing is safe or not, even if the truck in the middle of the road is blocking some camera views to observe the oncoming traffic. One extra camera on top of the traffic lights increases camera coverage. Table 5.5

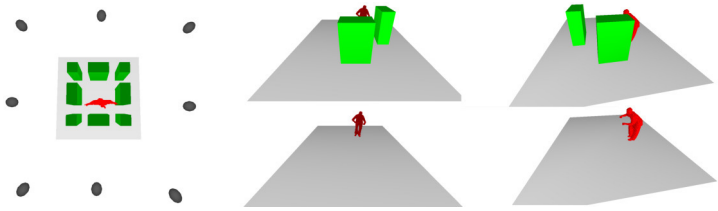


Figure 5.17: *Visualization of the simulated occluders. Each camera has one possible occluder (green) which can be either turned on or off. Two examples are shown The eight cameras are places facing the same area. All cameras are mounted at approximately 2.2m from the ground plane.*

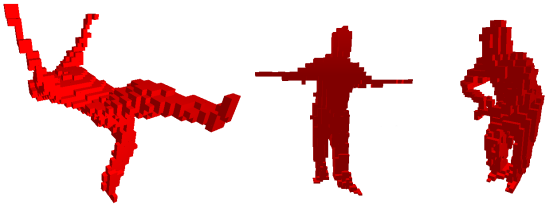


Figure 5.18: *Some examples from the reconstruction of the CVSSP-3D dataset [93] at 40 mm voxel size. In the experiments we use 20 mm.*

shows again that our methods performs almost as well as *Guan'06*.

Given the difficulty of automatically or manually labelling occluders in dynamic traffic scenes on which *Guan'06* relies, the good performance of the proposed method is a significant result. Even with severe occlusion, the algorithm manages to obtain a reliable reconstruction, as long as the object of interest can be partially observed from multiple viewpoints.

5.4.2 Experiment 2: qualitative comparison of the 3D reconstruction

The second experiment uses the JP sequences (breakdancer) from the CVSSP-3D dataset [93]. Each of these sequences consists of a synchronized stream of images from 8 cameras, which are placed around the subject at 2.2m high about every 45 degrees (see Figure 5.17). A total of six sequences is available. Each sequence is between 10 and 20 seconds long. In this experiment, we will simulate the presence of occluding objects and compare them with the ground truth, which is the output of the standard shape-from-silhouettes without occlusion (see Figure 5.18). Each frameset has equal weights on the evaluation. Therefore, displayed values are averages over each sequence. Reconstructions are performed at a cubic voxel size of 20 mm.

Method	Number of occluders							
	0	1	2	3	4	5	6	7
Laurentini'94	1.00	0.48	0.38	0.36	0.31	0.26	0.24	0.23
Guan'06	1.00	0.99	0.98	0.96	0.93	0.88	0.77	0.51
Landabaso'08	1.00	0.95	0.86	0.77	0.63	0.43	0.20	0.04
Slembrouck'17	1.00	0.99	0.97	0.94	0.90	0.81	0.64	0.30
$\beta = 3$	1.00	0.99	0.97	0.94	0.91	0.85	0.68	0.39
optimal β	1.00	0.99	0.97	0.94	0.91	0.85	0.69	0.39

Table 5.6: *Results in the case of multiple occluders in the scene. The occluders are placed as shown in Figure 5.17. The presence of each occluder creates an incomplete silhouette of the person. The best results are obtained by Guan'06, but this supervised method uses prior knowledge (occlusion masks where the occluders are nicely segmented). The proposed method performs best of all unsupervised methods. We show two results for our method: $\beta = 3$ and the optimal β per number of occluder based on the sensitivity analysis in Figure 5.20.*

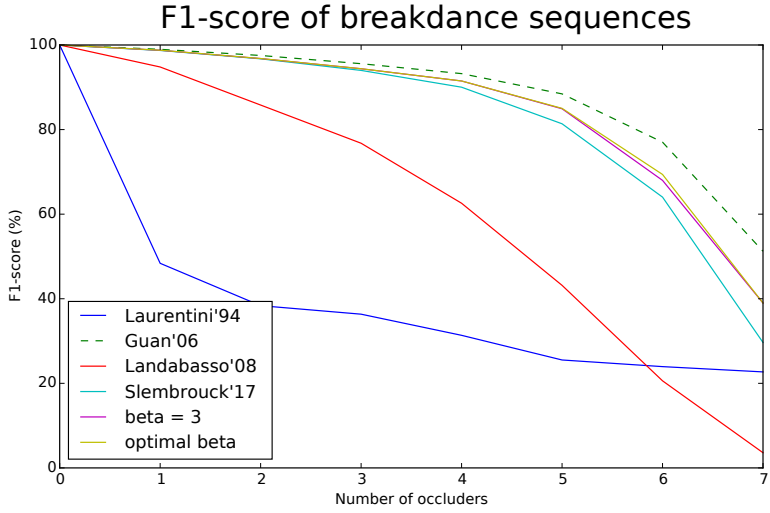


Figure 5.19: *Visual representation of the data in Table 5.6. The proposed methods performs best of all the unsupervised methods. We notice that the choice for β-value has a marginal impact on the performance. The proposed method performs slightly better than Slembrouck'17 because of its more sophisticated cost criterion.*

In this experiment, we simulate the presence of occluding objects by recalculating the input images as if there would be an occluder present. We defined an occluding object for each camera. This object is modelled as a vertical column partially blocking the camera's view (similar to

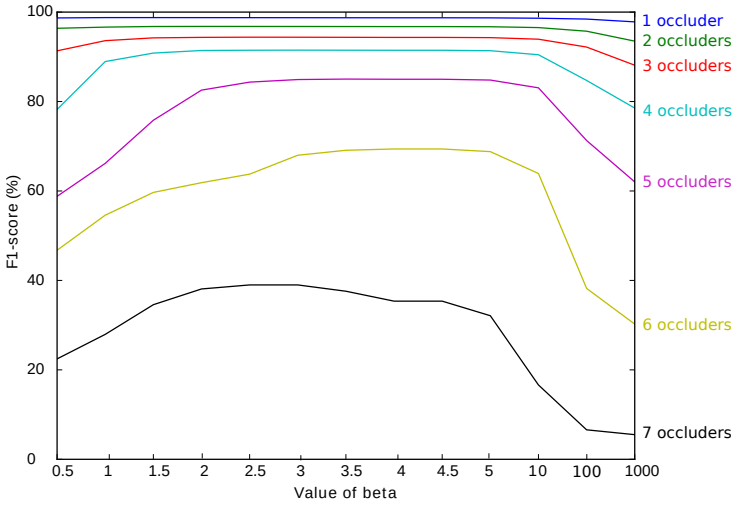


Figure 5.20: *Sensitivity analysis of the reconstruction of different β-values. β becomes more important in case of severe occlusion, but the range that produces suitable reconstructions is quite wide.*

a spectator in front of the camera. For each number between 1 and 8 occluders we choose eight random combinations (the same for each reconstruction method). For example, 2 occluders could block the view of camera 3 and 5. Each occluder roughly occludes 17% of the total image. We averaged the results over these combinations for each number of occluders. Figure 5.17 visualizes the camera setup together with the possible occluders.

In general, a lower value of β translates to a lower recall in the 3D reconstruction. On the other hand, lower values of β produce results with higher precision. Therefore the optimal value of β in terms of F1-score is neither zero nor very large. A suitable value for β should be empirically decided based on some training data during the setup of the system.

Table 5.6 and Figure 5.19 show the results for all methods, averaged over all sequences. Only the F1-score is shown, but for *Guan'06*, *Landabaso'08* and the proposed method, the recall is close or equal to 100%. Only *Laurentini'94* does not obtain a high recall value because occluded parts are not reconstructed. On the other hand, the precision of *Laurentini'94* is 100%. The last column represents the results of the proposed method.

In Figure 5.20 we provide a sensitivity analysis to assess the impact β on the reconstruction for a different number of occluders. The figure

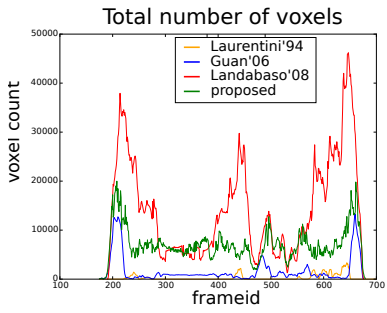
shows that the $F1$-score is large for a wide range of β, especially when the number of occluders in the scene is low. This optimal range of β decreases with an increasing amount of occlusion. Note that the scale in the X axis of the graph does not increase linearly near the end of the graph. In this setup, $\beta = 3$ seems a good choice.

5.4.3 Experiment 3: real-world single person tracking

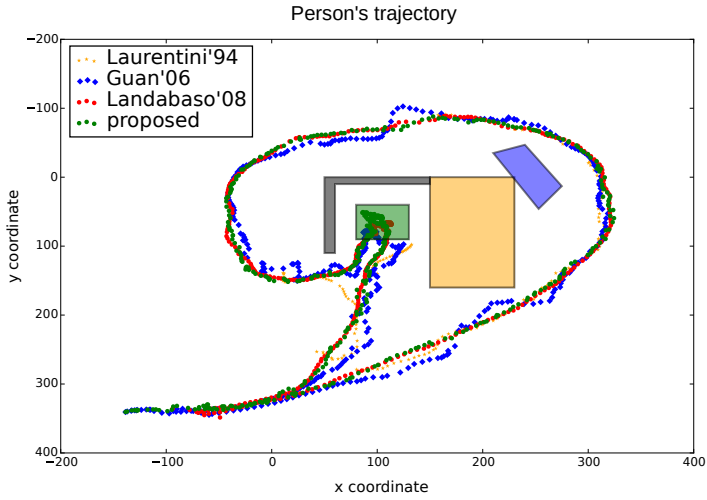
For the third experiment, we use the setup shown at the left in Figure 5.21a. It is a staged setup of an office environment. This setup has seven cameras mounted around the scene at about 3.5 meters high with a table, two chairs, an L-profile panel and a display introducing natural occlusion. The goal is to track the position of a person in the room. The plot in Figure 5.21c shows that the proposed method can track the person very well in the room. Note that the measured trajectories are not smoothed in any way. Although the recordings were made in a lab, the conditions were not optimal, especially for the foreground/background segmentation step because the colours of the clothes of the tracked person were similar to colours in the background, not unlike many real-world environments. A classical tracking algorithm suffers from tracking loss if the person is completely occluded for at least one of the cameras. Partial occlusion introduces inaccurate positions because the position is then calculated on a limited number of voxels.

Both the proposed method and *Landabaso'08* produce a person location in every frame from the moment the person enters the region of interest until he leaves again (whole track: 505 frames). On the other hand, *Laurentini'94* only outputs positions on 175 frames, which represents only 34.65% of the frames due to the lack of occlusion handling. The results of our method shows that occlusion handling improves the tracking results significantly. We also compared against *Guan'06* because that was the nearest competitor in the other experiments. We see that the track of this method is much less smooth than the one produced by our method. There are two reasons for that. *Guan'06* is more sensitive to erroneous foreground/background segmentation because it assumes complete silhouettes outside the known occluded regions (special regions) in each image. When a silhouette is still incomplete outside these special regions because of erroneous foreground/background segmentation, the shape reconstruction consists of fewer voxels and shows a rougher trajectory.

(a) office setup



(b) voxel count



(c) trajectory

Figure 5.21: *Visualization of the output from the proposed method, Guan'06, Laurentini'94 and Landabaso'08 in an office environment having multiple large occluders in the scene. The trajectories of the proposed method and Landabaso are the smoothest and most complete trajectories.*

Figure 5.21b shows the number of voxels that are part of the 3D reconstruction in each frame. We saw that *Landabaso'08* produced comparable tracking results, but the number of voxels varies significantly, which means the reconstructed shape of the person is less accurate. The reason why this is not visible in the tracking results is because the excess occurs in all directions in the 3D reconstruction and averages out when calculating the centroid of the reconstructed shape. When the reconstructed shape of the person needs to be analysed, this behaviour is unwanted, particularly when an accurate proximity detection is needed (e.g., cooperation between man and machine). In contrast, we see that the voxel count of the proposed method is much more stable.

5.5 Conclusion

This chapter explained different ways to represent occlusion and extend the standard visual hull algorithm effectively. Next to the manual assignment of occlusion in the form of masks or depth maps, we also presented an occlusion algorithm that automatically detects occluded parts and handles this correctly to complete a 3D reconstruction.

We presented an algorithm for shape-from-silhouettes that can cope with incomplete silhouettes. We showed that our algorithm performs well under different complexity levels of occlusion and without prior knowledge of the occluders. The algorithm automatically detects the occluded parts in the camera views and uses this information to reconstruct the object of interest, including the incomplete parts.

The algorithm succeeds in reconstructing the entire object of interest, and the reconstruction closely resembles the visual hull (i.e., the mathematical shape of an object based on the camera views as if there were no occlusion). As illustrated in the results section, our algorithm improves the state of the art for reconstruction as well as the tracking of a moving object in a scene with occlusion in an automated fashion.

The main contributions made in this chapter are the following:

- an extension of the standard shape-from-silhouettes algorithm with occlusion depth maps to handle static occluders where the occluders can be modelled per camera;
- a cell-based reconstruction method that can handle incomplete silhouettes originating from occlusion or silhouette segmentation errors.

Some of the work in this chapter is described in these publications:

- M. Slembrouck, D. Van Cauwelaert, D. Van Hamme, D. Van Haerenborgh, P. Van Hese, P. Veelaert, and W. Philips. Self-learning voxel-based multi-camera occlusion maps for 3d reconstruction. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP'14)*, page 8. SCITEPRESS, 2014
- M. Slembrouck, D. Van Cauwelaert, P. Veelaert, and W. Philips. Shape-from-silhouettes algorithm with built-in occlusion detection and removal. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP'15)*. SCITEPRESS, 2015

- M. Slembrouck, J. Niño Castañeda, G. Allebosch, D. Van Cauwelaert, P. Veelaert, and W. Philips. High performance multi-camera tracking using shapes-from-silhouettes and occlusion removal. In *Proceedings of the 9th International Conference on Distributed Smart Cameras (ICDSC)*, pages 44–49. ACM, 2015. ISBN 978-1-4503-3681-9

- J. Niño Castañeda, A. Frias Velazquez, B. B. Nyan, M. Slembrouck, J. Guan, G. Debard, B. Vanrumste, T. Tuytelaars, and W. Philips. Scalable semi-automatic annotation for multi-camera person tracking. *IEEE Transactions on image processing*, 25(5):2259–2274, 2016. ISSN 1057-7149

- M. Slembrouck, J. Niño Castañeda, G. Allebosch, D. Van Cauwelaert, P. Veelaert, and W. Philips. High performance multi-camera tracking using shapes-from-silhouettes and occlusion removal. In *Proceedings of the 9th International Conference on Distributed Smart Cameras (ICDSC)*, pages 44–49. ACM, 2015. ISBN 978-1-4503-3681-9

- M. Slembrouck, P. Veelaert, D. Van Hamme, D. Van Cauwelaert, and W. Philips. Cell-based approach for 3d reconstruction from incomplete silhouettes. In *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS'17)*. Springer, 2017

- M. Slembrouck, P. Veelaert, D. Van Cauwelaert, D. Van Hamme, and W. Philips. Cell-based shape reconstruction from incomplete silhouettes. *Integrated Computer-Aided Engineering*, 26(3):257–271, 2019. ISSN 1069-2509

Chapter 6

3D Human Pose Estimation

The 3D reconstruction explained in the previous chapters already allows several analyses, such as an approximated calculation of the centre of mass, the volume and the occupied space. For a reliable biomechanical analysis of human body poses and human motion analysis we also want to reconstruct the person's pose in 3D. Typical application domains are gaming, rehabilitation, sports, and general behaviour analysis.

In this chapter, we will first illustrate an attempt to reconstruct the 3D pose of a basketball player from the volumetric shape during basketball free-throws. Because recent advances in pose estimators in images offer reliable pose estimators with which a person's pose can be analysed in detail, we abandoned this track and focused on the reconstruction of the 3D pose using 2D poses. These pose estimators are driven by a neural network and are usually capable of detecting poses of multiple people in a single image. We extend the single image pose analysis to multi-view pose analyse to reconstruct 3D poses.

The 3D reconstructed pose is tracked over time to perform specific application-dependent analysis. We tackle many problems such as the cross-view pose correspondence problem and the problem that some keypoints are not correctly detected (either occluded, poorly detected, out of view or not detected by the pose estimator). In this chapter we first discuss different neural networks capable of detecting keypoints in 2D images. Second, we explain how we obtain 3D poses. Third, we discuss how we solved the cross-view pose correspondence problem to reconstruct multiple poses in the scene. Fourth, some experiments illustrate qualitative results where we compare our technique against marker-based systems, which are the golden standard for reconstructing human poses.

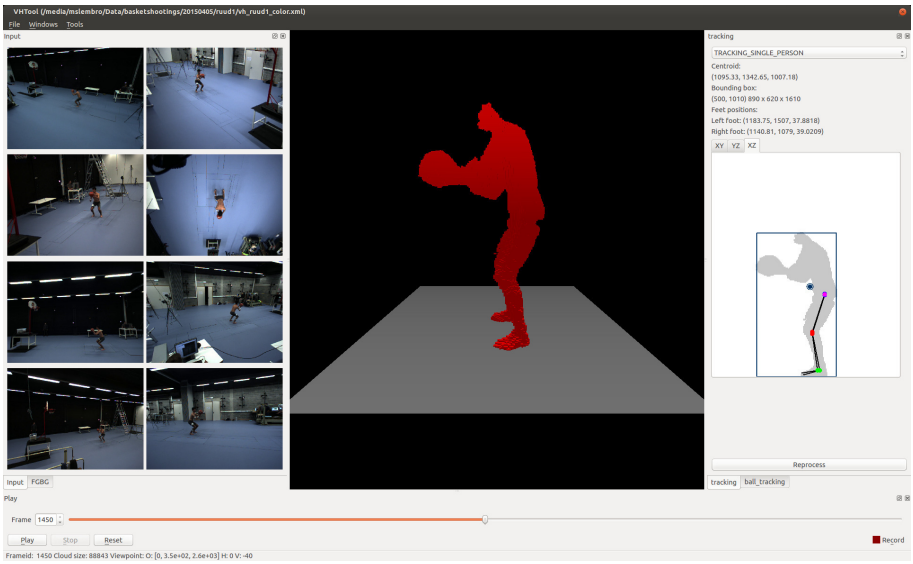


Figure 6.1: *System overview of the shape-based pose estimation in basketball free-throws. The program visualizes the input images (left), 3D reconstruction (center) and keypoint model (right).*

6.1 Pose estimation from 3D shapes

Volumetric shapes from humans can be used to fit a 3D keypoint model inside. However, such an approach relies on assumptions which are only met in specific circumstances. As a starting point for this research track, we decided to analyse basketball free-throws in cooperation with the Department of Movement and Sport Sciences of Ghent University.

Basketball free-throws form an interesting use case, because the movement is well-defined and numerous aspects of a human pose can be analysed during the different phases of the free-throw. Each phase requires the proper technique including the position of key body parts: footwork, lower body, upper body, head, arm, hand [20].

Based on eight calibrated camera views, a reliable shape reconstruction was created using the techniques from the previous chapters. Figure 6.1 shows the visualization of the program that was developed for this use case with input images (left), the reconstruction (middle) and an early attempt at extracting a pose of the lower body (right). This basic keypoint model consists of eight keypoints: the foot tip, the ankle, the knee and the hip from both side of the body. We extracted these keypoints using the following constraints: each connection between the skeleton's keypoints has a fixed length and the keypoint is located in

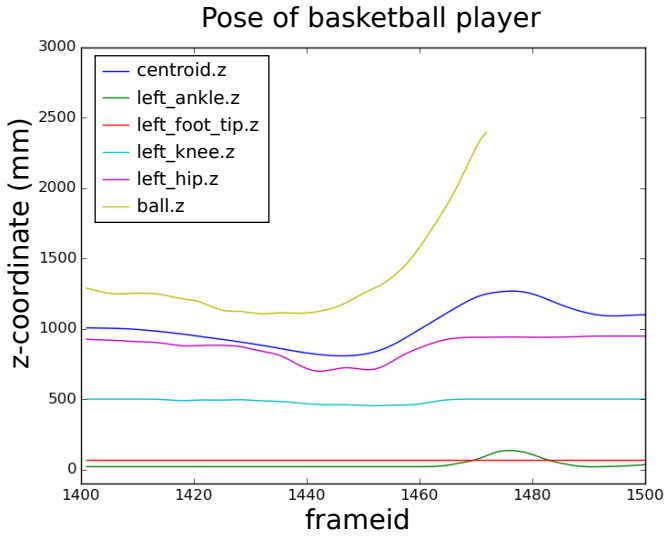


Figure 6.2: *Temporal analysis of several keypoints of the simplified keypoint model and the ball during a free throw.*

the centre of the shape. We also assume that the player is facing the hoop so that the left and right side of the player is easily identified. The reconstruction starts from the foot tips and works its way up to the hip.

The pose estimation in Figure 6.2 temporarily tracks Z-coordinates (vertical direction) of six points in time: the centroid, the left foot tip, the left ankle, the left knee, left hip and the ball. The right leg was also tracked but omitted from the graph for clarity because the movement is symmetrical during this free-throw. The ball was tracked using colour segmentation and volumetric shape reconstruction. The ball is lost once it exits the reconstruction volume. However, the focus was on the basketball player, rather than the ball.

Under these controlled circumstances where the movement is known, the lower body tracking is reliable enough to analyse the motion. However, for a complete analysis of the basketball free-throw, reliable upper body tracking is also required and it was not possible to create this based on the shape reconstruction only. That is mainly because the upper body, the arms and the ball could not be easily separated in the 3D reconstruction when the ball was close to the player's body.

For more precise and reliable 3D human pose estimation, we explored 2D pose estimators. The big advantage of these estimators is that they detect several keypoints that already form poses in 2D, which is a completely different approach than fitting a 3D keypoint model

into a 3D shape. Pose estimators are also more reliable because each detected keypoint can be used as an anchor point for the 3D pose. In the remainder of this chapter, we therefore focus on 3D pose estimation based on 2D pose estimators.

6.2 Pose estimation in 2D images

6.2.1 Artificial neural network

An artificial neural network (ANN) is a machine learning technique consisting of simple interconnected processing units. Such an ANN is designed to simulate the way the human brain analyses and processes information. ANNs learn by example, and, in general, better results can be expected when more training data is available. However, the training data should not be overspecific to avoid over-fitting, which means that the ANN learns particular oddities specific to the training data and therefore performs significantly worse on unseen data.

We use pre-trained neural networks. Therefore, the focus is on the outputs of these networks instead of efforts to improving the networks themselves. More specifically, the detected image coordinates (keypoints) are used to estimate the 3D location of each of these keypoints using multiple camera views, while handling possible erroneous positions and labels.

6.2.2 Neural networks for pose estimation

In this work, we are interested in tracking keypoints from the human body in 3D. Most of these keypoints represent joints such as the knee joints and ankle joints. Unlike a mechanical joint, the position of most joints in the human body are ill-defined because such a joint is encapsulated in different tissue layers (invisible to an RGB camera), and the rotation centres are not positioned in fixed places (because the joints are not perfect spheres). Therefore, these ANNs learn the keypoint's most likely position based on a large number of samples. The keypoints are not detected in the exact anatomical position, but we noticed that the detected positions are consistent in different frames. It means that the same 3D keypoint is detected in consecutive frames. From the experiments we noticed that an offset in position between the detected keypoints and the position determined by maker-based systems exists. However, the important thing is that the detected keypoints can be

followed reliably over time so that they move in the same way as the corresponding anatomical joint. From these consistently moving keypoints it is possible to extract a person's pose, which is the main objective of this work. In the experiments section (see Section 6.6), we will show that that is indeed the case.

The topic of multi-person pose estimation from a single camera view has gained renewed attention thanks to the development of convolutional neural networks. However, inferring the pose of multiple people in images presents a unique set of challenges:

1. Each image contains an unknown number of people that can appear at any position or scale.
2. People may not be fully visible, or multiple people in an image may occlude each other, making it hard to assign detected body parts to the correct person.
3. Runtime complexity tends to grow with the number of people in the image, making real-time performance a challenge.

In the remainder of this chapter, these challenges will be overcome to create a real-time multi-person 3D pose estimation system.

I. Popular pose estimators

This section gives an overview of some of the more popular monocular pose estimators used nowadays. The use case is a markerless human posture recognition system that can be used for rehabilitation/athlete performance measuring and game control. Each application aims to track the a person's pose precisely. In rehabilitation or athlete performance, precision is critical, while the real-time processing component is less important. For game control, real-time processing is essential, and results may be slightly less precise.

The extraction of poses from single views is not trivial. First, each image may contain an unknown number of people that can occur at any position or scale. Second, interactions between people induce complex spatial interference due to contact, occlusion, and limb articulation, making the association of parts difficult. Third, runtime complexity tends to grow with the number of people in the image, making real-time performance a challenge.

The design and testing of a pose estimator take years to optimise. Therefore, existing pose estimators are used in this work, and choices are made based on processing speed, accuracy, and a number of relevant keypoints that the pose estimator can detect. For instance, only a limited number

of pose estimators provide keypoints on the feet (important for tracking running motion). These pose estimators are OpenPose with the BODY_25 keypoint model [16] and VNect [64]. We will discuss these two first and mention alternatives afterwards.

OpenPose and VNect State-of-the-art pose estimation from a single view consists of two main approaches which rely on detections of 2D keypoints in the image. However, their focus may be different because either 2D poses (OpenPose) or 3D poses (VNect) might be estimated from these detections.

The first approach uses keypoint detections in 2D and creates a 2D poses. OpenPose is one of the most popular 2D real-time multi-person pose estimators at the moment [16, 15, 102] because not only the code is opensource (under license for commercial use), but also the network and trained weights are provided. Unlike other approaches, OpenPose uses a bottom-up approach by first detecting keypoints of body parts and consecutively assigning those to persons (more info can be found in section II.).

The second approach estimates a 3D pose directly from a single image, which in fact is also the aim of this chapter, but using multiple views to solve the ambiguities which are typical for monocular solution. The leading method in that respect is VNect [64]. VNect claims comparable or even better results than reconstruction from RGB-D data generated by the Microsoft Kinect [82]. Although the detection are in 3D, this method can still be used as input to the proposed pose estimation method by using the corresponding 2D locations of the detected keypoints.

Other pose estimators Besides VNect and OpenPose, some other pose estimator frameworks also exist. The applications we have in mind require reasonable processing speed, which means that the pose estimator should guarantee at least 15 fps.

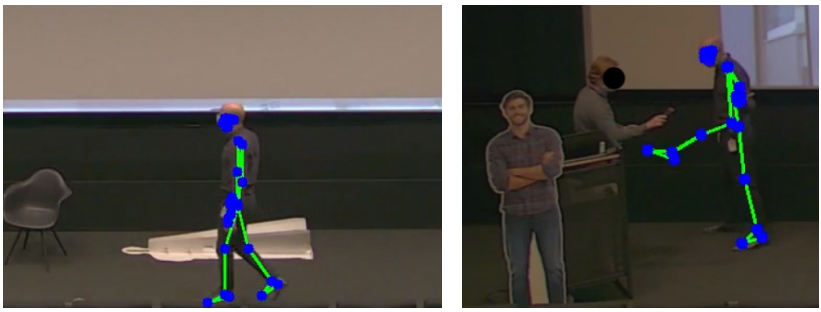
DeepCut [75] and DeeperCut [46] are two pose estimators that perform well. These two methods use bottom-up proposals for body parts but require costly global inference to obtain joint association, which significantly impacts the computational performance. Both extractors report a minimum processing time of 230 seconds per frame, which is too slow for real-time applications. The number of supported keypoints is also limited to 14 keypoints. Therefore, it is impossible to track certain aspects of the human pose, such as the pose of the feet or the head orientation.

PoseFlow and AlphaPose by Shanghai Jiao Tong University [110] offer similar performance like OpenPose when there only one pose per view can be found. However, unlike OpenPose, the computation time linearly increases with the number of people in an image, which creates issues in real-time processing because a constant execution time per frame cannot be guaranteed. The linear processing time for each detected person is due to the top-down approach where a people detector first detects the people in the image, and then the keypoints are searched within the detected bounding box. Although many applications in this work target single person reconstruction, there are often multiple people in view by different cameras e.g., bystanders, controllers, etc.

The commercial system wrnchAI (by wrnch) [60] seems to outperform OpenPose in terms of processing speed (about 2.5 times faster). The overall performance of both wrnchAI and OpenPose is similar, although OpenPose handles false positives better than wrnchAI. The GPU memory requirement is significantly lower for wrnchAI (1 GB) compared to OpenPose (2.5 GB), which is an advantage when the network has to run on less powerful devices like smartphones. There is no information to be found on how wrnchAI exactly works.

In the Detectron2 framework [104], Keypoint R-CNN is included, which is a variant of Mask R-CNN (see Chapter 3) to solve keypoint detection tasks [41]. The method uses the COCO keypoint model which supports 18 keypoints. Keypoint R-CNN is available under the Apache 2 license which allows it to be used for commercial use. However, the performance is lower than that of OpenPose. Keypoint R-CNN might however be a suitable candidate for commercial use cases where the extra keypoints from OpenPose are not necessarily required.

Recently, BlazePose was presented in [8]. BlazePose uses a lightweight CNN that runs at high frame rates on CPUs. Frame rates up to 50 fps are achieved on a 640x360 px image on an Intel Core i5 @ 4 x 3.5 GHz. Even on mobile phones, BlazePose reaches 30 fps (Pixel 2 phone). CPU-based approaches have a significant advantage over the other heavy networks because they do not require expensive GPU cards. Moreover, BlazePose detects 33 body keypoints which is the highest number of keypoints at such frame rate. OpenPose is the absolute winner in that respect with 25 body keypoints, 2 x 21 hand keypoints, and 70 face keypoints. However, the hand and face detections rely on the body keypoints and cause a significant delay in processing speed, unlike BlazePose, which simultaneously detects all keypoints. Unfortunately, BlazePose is limited to a single person pose detection, and the reliability is coupled to the relative size of the person in the image (Figure 6.3).



(a) correct pose *(b) erroneous pose*

Figure 6.3: *Some results from BlazePose are shown. This CPU-powered real-time pose reconstruction performs well as long as there is only one person in view and the person can be clearly recognized. The method fails when either of those requirements is not fulfilled as seen in (b).*

When multiple people are present in the scene, BlazePose only detects one (or none) and uses tracking to avoid switching between different people in a video. The most significant issues arise when people are close together because the keypoints from multiple people are mixed into a single erroneous pose. Blazepose also makes mistakes when the person cannot be easily recognised. For instance, in Figure 6.3b, Blazepose detects a rather unlikely pose. For certain specific single-person use cases, BlazePose might still be suitable.

Note that various companies provide frameworks to speed up training and inference of pose estimation, in some cases providing pre-trained pose estimation networks such as NVidia Human Pose estimation with Deepstream SDK [80], Intel OpenVINO toolkit [71] and Xilinx Edge AI Solution [109].

Choice for our application Since our application aims to track the complete movement of the human body as precisely as possible, we must track all body parts. Therefore, the number of supported keypoints and their positions on the human body has a significant impact on our decision. With the BODY_25 model in OpenPose, 25 keypoints are supported, whereas VNect is more limited: only 17 keypoints. The extra keypoints are located at the end of the feet, the neck, eyes and ears. Thus, OpenPose detects the feet positions more accurately and also facilitates measuring other relations in the pose. For instance, the angle between the foot and the corresponding lower leg is important for gait analysis, an experiment performed in collaboration with the Department of Movement and Sport Sciences at Ghent University in the

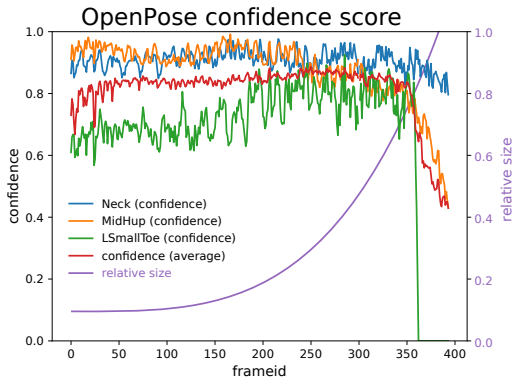
Sport Science Laboratory Jacques Rogge (IRUNMAN project). Also, we do not need the 3D estimation that VNect provides since we will calculate the estimated 3D positions from multiple cameras, which allows us to cope with occluded keypoints. OpenPose also processes data faster than VNect, which is essential in real-time applications such as gaming.

II. OpenPose: Realtime multi-person 2D pose estimation

There is a reason why OpenPose [16] is the preferred pose estimator to obtain real-time performance. In contrast to the top-down approaches discussed above, OpenPose handles the multi-person 2D pose estimation differently. OpenPose uses a bottom-up approach where the keypoints are first detected, and in a second phase, these keypoints are assigned to persons. Bottom-up approaches do not scale linearly with the number of people in the image because the keypoints are detected in parallel in the first phase, while top-down approaches first detect the different people in the image and analyse each detection in a second phase. Furthermore, bottom-up approaches are more robust against choices made in early phases because the network searches for the different keypoints independently. In contrast, the performance of a top-down approach depends on whether or not a person has been detected, which often fails when multiple people are in close proximity. Early attempts at bottom-up approaches did not offer the gain in efficiency because the keypoint association requires costly global inference [75, 46]. OpenPose simultaneously infers bottom-up representations of detection and association and shows that global context is encoded sufficiently well to allow a greedy pass to achieve high-quality results at a fraction of the computational cost, resulting in a reliable real-time pose estimator.

We will not go into too much detail about the OpenPose framework because we treat it as a black box. However, we discuss some of its features as a good practice guide. In general, neural network detectors perform better when the object is significantly large in the image. OpenPose is no exception to this. The network aims to find 25 keypoints for which the person should take up a considerable part of the image. We aim to make sure that the person's size is at least 20% of the height of the image. A person is best detected when fully seen in the image. When too close to the cameras, some keypoints are out of view, which can impact the detected pose. When too few keypoints are in view, OpenPose will not even recognise this subset of keypoints.

A neural network is as strong as the training data. The OpenPose network was trained on image rotation of $\pm$ 40 degrees. Therefore,



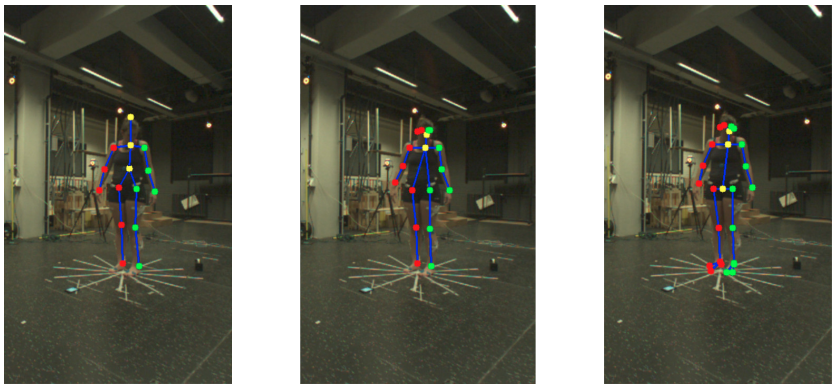
(a) confidence scores / relative size

(b) input images

Figure 6.4: *OpenPose confidence scores in relation to relative size of a person. A test subject runs towards the camera, starting from about 75 m away (fixed zoom factor). In (a) we notice that the confidence scores of the main keypoints are not impacted significantly. Detailed keypoints such as the left small toe (LSmallToe) we notice an increase in confidence when the test subject appears larger in the image. When the test subject is no longer completely in the field of view of the camera, the confidence scores drop. In (b), we show the detected poses, roughly every 15 m.*

OpenPose is unable to detect a person that appears upside down in the image. Sometimes, cameras are mounted upside down, or a $\pm$ 90 degrees rotation because of practical reasons. An easy fix to make OpenPose perform optimally is to rotate the image according to the main axis of the person in the view: 90, 180 or 270 degrees and process the images accordingly. This issue could be solved by retraining with people in more orientations, but results in highly reduced accuracy. Rotating the image is undoubtedly a better option. For the same reason, OpenPose has difficulty recognising people in non-conventional positions, e.g., a breakdancer doing a flip, because such poses were not part of the training data.

For each keypoint, OpenPose reports a position and a confidence score. These confidence scores originate from a confidence map highlighting the likelihood of a keypoint appearing in a certain region. Hence, these values are related to the reliability of a detected keypoint. We conducted an experiment to analyse the relation between the size of the person in the frame and the confidence scores. Figure 6.4 summarises the experiment. In general, the confidence scores increases when the size of the person becomes larger. However, detailed keypoints, such as the LSmallToe increase significantly when the size of the person increases.



(a) MPI *(b) COCO* *(c) BODY_25*

Figure 6.5: *Three available keypoint models in OpenPose in the IPLAY Leuven dataset. The BODY_25 dataset offers the most keypoints.*

The Neck keypoint is almost not impacted by size in terms of confidence. The MidHip shows similar behaviour with the exception of the images where the person is very large (and not fully visible in the image). Intuitively it makes sense that the MidHip's position is more confident from a distance then when the person is closeby (large in the camera). The size of the person is just one variable to analyse, there might be other reasons why the score is low, such as occlusion (see Section 6.5.1) and limb ambiguity (see Section 6.5.2).

OpenPose supports three keypoint models. In Table 6.1 we list all of the keypoints from the different models in OpenPose that are currently supported (also shown in Figure 6.5). The BODY_25 model supports the most number of keypoints. The extra keypoints are mainly located in the lower part of the body (feet) and includes an extra keypoint for the MidHip, which is very useful in combination with the Neck keypoint to determine the centreline of a person.

III. Alternative keypoint models

The proposed neural networks are not limited to the detection of a person's joints. Similar neural networks can also be trained to detect hand keypoints or facial landmarks. The OpenPose framework, for instance, supports 21 keypoints per hand (one for the hand palm and 4 additional keypoints for each finger) and 70 keypoints for the facial landmarks. Note that the hand sizes and face size needs to be large enough to detect these detailed keypoints. To facilitate hand detection,

Keypoint	**MPI**	**COCO**	**BODY_25**
Head	x (0)		
Nose		x (0)	x (0)
Neck	x (1)	x (1)	x (1)
RShoulder	x (2)	x (2)	x (2)
RElbow	x (3)	x (3)	x (3)
RWrist	x (4)	x (4)	x (4)
LShoulder	x (5)	x (5)	x (5)
LElbow	x (6)	x (6)	x (6)
LWrist	x (7)	x (7)	x (7)
MidHip			x (8)
RHip	x (8)	x (8)	x (9)
RKnee	x (9)	x (9)	x (10)
RAnkle	x (10)	x (10)	x (11)
LHip	x (11)	x (11)	x (12)
LKnee	x (12)	x (12)	x (13)
LAnkle	x (13)	x (13)	x (14)
Chest	x (14)		
REye		x (14)	x (15)
LEye		x (15)	x (16)
REar		x (16)	x (17)
LEar		x (17)	x (18)
LBigToe			x (19)
LSmallToe			x (20)
LHeel			x (21)
RBigToe			x (22)
RSmallToe			x (23)
RHeel			x (24)

Table 6.1: *Keypoints of the OpenPose models with keypoint IDs.*

OpenPose uses regions of interest around the detected wrist keypoints and detects the hand points only inside these regions of interest to reduce computation time and make the detections more reliable. In this work, hand detection was used to assess hand movements in more detail to detect pick actions in the COSMO use case where a first aid kit is assembled (see Figure 6.6). When the operator is holding a piece part, fewer keypoints are found due to occlusion, which is a reliable indicator to classify grab actions.

Figure 6.6: *COSMO first aid kit assembly use case with hand detection.*

6.3 Triangulation

Traditional cameras observe the 3D world by light ray projection on a 2D plane. During this process, depth information is lost, which is why it is hard to reconstruct a 3D scene from a single captured image accurately. With multiple cameras with differing viewpoints, 3D reconstruction is possible provided calibration data of the different cameras is available [40].

The mathematical conversion of 2D points from multiple cameras into a 3D location is often referred to as triangulation. The idea is to estimate the position of point $\boldsymbol{r}$ based on the 2D image positions $\widetilde{\boldsymbol{u}}_j$ (Figure 6.7). Due to inaccuracies in the camera calibration and the discretisation of the image sensor, the different lines through each c_j and $\widetilde{u}_j$ will rarely intersect. Also, triangulation of a 3D point relies heavily on the assumptions that the 2D point correspondences in multiple views belong to the same 3D point. In this case, the 2D keypoints are labelled by the pose estimator, and we know which 2D points should be combined. However, misdetections and errors in the 2D measurement will still impact the estimated 3D position of the corresponding point. Therefore, outliers are best removed from the set of 2D points if possible.

In this section, we study the two main approaches in more detail. The most common approach is to find a 3D point that minimises the sum of the squared reprojection errors for all observations of a certain point. Another approach is to minimise the Euclidean distance between lines where the camera's focal point and the corresponding observation point defines each line. The latter approach searches the midpoint between all these lines.

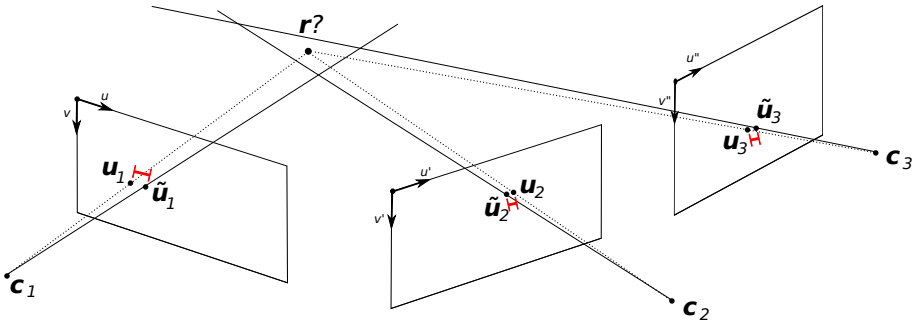


Figure 6.7: *Triangulation example where we need to estimate the 3D position of point* $\boldsymbol{r}$ *based on 2D points from multiple cameras. The reprojection errors are indicated. Ideally, both* $\boldsymbol{u}_j$ *and* $\widetilde{\boldsymbol{u}}_j$ *would coincide.*

I. Minimizing the reprojection error

Figure 6.7 illustrates the triangulation problem based on reprojection errors. The projection matrix P_j determines the image location of a 3D point on the image sensor of camera j (see Chapter 2). The unknown 3D location of point $\boldsymbol{r}$ is observed by camera j as a point with homogeneous pixel coordinates $\boldsymbol{u}_j(\boldsymbol{r}) = P_j\boldsymbol{r} = [sx_j(\boldsymbol{r}), sy_j(\boldsymbol{r}), s]^T$. Due to noisy observations however, each camera j observes this point at a slightly different location $\widetilde{\boldsymbol{u}}_j = [\widetilde{x}_j, \widetilde{y}_j, 1]$. The Euclidean distance between $\boldsymbol{u}_j(\boldsymbol{r})$ and $\widetilde{\boldsymbol{u}}_j$ is defined as the reprojection error for camera j [39] and can be expressed as:

$$\gamma_j(\boldsymbol{r}) \triangleq \|\boldsymbol{u}_j(\boldsymbol{r}) - \widetilde{\boldsymbol{u}}_j\| = \sqrt{(x_j(\boldsymbol{r}) - \widetilde{x}_j)^2 + (y_j(\boldsymbol{r}) - \widetilde{y}_j)^2}. \tag{6.1}$$

Such an error can be calculated for every camera having an observation $\widetilde{\boldsymbol{u}}_j$. As such, a vector is constructed which contains all reprojection errors: $\gamma(\boldsymbol{r}) = [\gamma_1(\boldsymbol{r}), \gamma_2(\boldsymbol{r}), \ldots, \gamma_N(\boldsymbol{r})]$. For a certain set of observations $(\widetilde{x}_j, \widetilde{y}_j)$, the values in $\gamma(\boldsymbol{r})$ change when the 3D location of $\boldsymbol{r}$ varies. The goal is to find the point $\boldsymbol{r}$ for which a cost over all elements in $\gamma(\boldsymbol{r})$ is minimal. Often, the norm of the vector is used as a cost function. The most common are the ℓ_1 norm (sum of magnitude), ℓ_2 norm (sum of squares) and ℓ_∞ norm (maximum) of image reprojection errors.

The ℓ_2 and ℓ_∞ norm are often used in the context of triangulation. Although outliers have the most impact on the ℓ_∞ norm compared to the ℓ_2 norm, the ℓ_∞ is significantly simpler than the ℓ_2 cost. Moreover, ℓ_∞ minimisation involves finding the minimum of a cost function with a single local (and hence global) minimum on a convex parameter domain, while the ℓ_2 cost function has multiple minima.

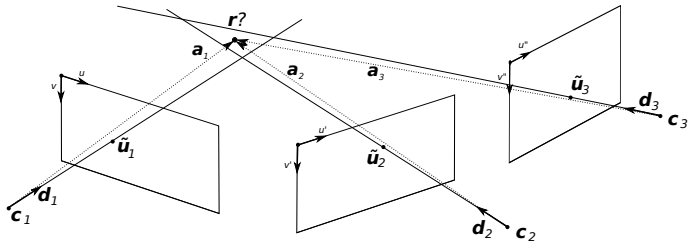


Figure 6.8: *Triangulation example where we need to estimate the 3D position of point* $\boldsymbol{r}$ *based on 2D points from multiple cameras using the midpoint approach. Ideally vectors* $\boldsymbol{a}_i$ *and* $\boldsymbol{d}_i$ *would coincide.*

II. Finding the midpoint

The previous approach minimises the reprojection error for each of the camera views (2D measurement). At the same time, the problem can also be solved in 3D by searching for the midpoint of the lines defined by the camera positions and the corresponding estimates $\widetilde{\boldsymbol{u}}_j$ (Figure 6.8) [9]. The 3D world point $\boldsymbol{r}$ is the point for which the sum of the squared distances between that point and each line is minimal (ℓ_2 norm). Each such a line is defined by the focal point $\boldsymbol{c}_j$ of camera j and unit vector $\boldsymbol{d}_j$ which is defined by $\boldsymbol{c}_j$ and $\widetilde{\boldsymbol{u}}_j(r)$. In order to formulate the problem, we also define the vector $\boldsymbol{a}_j = \boldsymbol{r} - \boldsymbol{c}_j$.

The point to line distance (distance between $\boldsymbol{r}$ and the line $\boldsymbol{c}_i + \lambda \boldsymbol{d}_i$, with $\lambda > 0$) is then given by the ℓ_2 norm of $\boldsymbol{w}_j$:

$$||\boldsymbol{w}_j||_2 = \sqrt{\boldsymbol{w}_j \cdot \boldsymbol{w}_j} \text{ where } \boldsymbol{w}_j = \boldsymbol{d}_j \times \boldsymbol{a}_j. \tag{6.2}$$

We now determine the single 3D point $\boldsymbol{r}$ that minimizes the sum of squared point to line distances $\sum_j ||\boldsymbol{w}_j||^2$. This minimum occurs where the gradient is the zero vector ($\mathbf{0}$):

$$\nabla \left(\sum_j ||\boldsymbol{w}_j||^2 \right) = \mathbf{0}.$$

Expanding the gradient,

$$\sum_j (2\boldsymbol{d}_j(\boldsymbol{d}_j \cdot \boldsymbol{a}_j) - 2(\boldsymbol{d}_j \cdot \boldsymbol{d}_j)\boldsymbol{a}_j) = \mathbf{0}.$$

It follows that the coordinates of $\boldsymbol{r}$ satisfy a 3x3 linear system,

$$M\boldsymbol{x} = \boldsymbol{b}, \tag{6.3}$$

Figure 6.9: *Midpoint between two almost parallel lines are may lead to a singular matrix M. Multiple camera views at sufficiently different viewpoints solve this problem.*

where the kth row (a 3-element row vector) of matrix M is defined as

$$M_k = \sum_j (d_{jk}\boldsymbol{d}_j - (\boldsymbol{d}_j \cdot \boldsymbol{d}_j)\boldsymbol{e}_k)^T$$

with vector $\boldsymbol{e}_k$ the respective unit basis vector, and

$$\boldsymbol{b} = \sum_j \boldsymbol{d}_j(\boldsymbol{c}_j \cdot \boldsymbol{d}_j) - \boldsymbol{c}_j(\boldsymbol{d}_j \cdot \boldsymbol{d}_j).$$

In rare circumstances, M may be singular, or the value of its determinant may be close to zero (see Figure 6.9). For example, we could have a system with only two cameras facing each other and a point close on the line joining the two projection centres. Since this situation can be easily avoided, we exclude it because we chose the camera setup to limit the occurrences of this issue and usually, more than two cameras are observing the same point.

6.4 Cross-view pose correspondence

The reconstruction of poses observed from different views is straightforward when only one pose is present in each view and all poses belong to the same person. However, this constraint is rarely met in the real world. Even in lab settings, cameras may observe multiple people, of which some are visible in the background. Therefore, it is essential to find the cross-view pose correspondences of the poses detected in the different views. If poses from different people are triangulated, the resulting pose is incorrect and forms a so-called ghost reconstruction.

However, matching 2D poses across multiple views is challenging. A typical approach is to use the epipolar constraint to verify if two 2D poses are projections of the same 3D pose for each pair of views [47]. Nevertheless, this approach may fail for the following reasons. First,

camera	# 2D poses
C1	2
C2	3
C3	3
C4	2

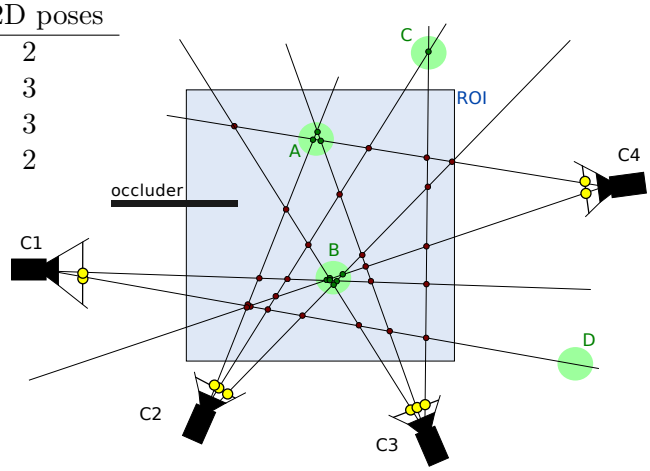


Figure 6.10: *Cross-view pose correspondence problem: The table indicates the number of detected poses in each view. Persons A, B and C are observed by more than one camera, person D only by one camera. Persons C and D are located outside the region on interest (ROI).*

the detected 2D poses are often inaccurate due to heavy occlusion and truncation, making geometric verification difficult. Second, matching each pair of views may produce inconsistent correspondences, which violate the cycle consistency constraint. That is, two corresponding poses in two views may be matched to different people in another view. Such inconsistency leads to incorrect multi-view reconstructions. Finally, different sets of people appear in different views, and the total number of people is unknown, which brings additional difficulties to the matching problem.

The correct correspondences are not trivial to find due to occlusion, people out of view for particular camera views and inaccurate keypoint detections. To illustrate the significant difficulties, let us consider a simple camera network with four cameras in Figure 6.10, where the neck keypoint of each pose is backprojected as a line. In total, there are four people present in this example, indicated as big green circles A, B, C and D. Looking at the observations (yellow circles) of the cameras, we see that the cameras C1, C2, C3 and C4, respectively, observe 2, 3, 3 and 2 poses. Not all four persons are detected because poses are out of view, e.g., person D for cameras C2, C3 and C4, or an occluder prevents the observation, e.g., person A in camera C1.

Many line intersections can be observed in Figure 6.10, each of which may correspond to a correctly or incorrectly triangulated point. In

general, when at least three cameras observe a person, a cluster of intersection points will emerge near the actual 3D position of the person. However, looking at the bottom left intersection, this not necessarily means that a cluster always corresponds to a person. The example shows that the people correspondence problem is not trivial. In essence, the pose observations from the different views need to be matched with each other. However, the number of people in the scene is unknown.

Now that the problem has been explained, we focus on the solution. In literature, this particular correspondence problem is solved in two ways. The most popular approach is using a 3D pictorial structure (3DPS) model that implicitly solves the correspondence problem by reasoning about all hypotheses in 3D that are geometrically compatible with 2D detections. The other approach adds appearance models for the detected people and matches people appearances in other views. Both approaches have the aim to find clusters of pose detections from different views to calculate the 3D pose. The latter approach claims to improve the first approach because the appearance model drastically reduces the number of good correspondences. However, when people have a similar appearance, this approach may fail. Calculating appearance models also adds computation time, and occlusions may introduce errors because a person might only be partly visible, while the bounding box of two detected persons may overlap. Moreover, the number of people in our applications is limited.

3DPS-based approaches are computationally expensive due to the enormous state space (different people each having 25 different keypoints to match with an unknown number of people). This state space is reduced by introducing a two-phase algorithm where the first phase aims to couple poses from different views in a pairwise manner. In contrast, the second phase handles the clustering and triangulation of poses in more than two views.

6.4.1 Pairwise correspondences

The goal is to assign all 2D poses to the corresponding poses in the scene and triangulate their position. The number of combinations can be calculated as follows: from each view, we can choose one of the poses, or none of the poses, leading to $n_j + 1$ possibilities per view where n_j represents the number of poses in that view. The number of poses in a combination varies between 0 and N. The total number of combinations is then $\prod_{j=1}^{N}(n_j + 1)$. However, a valid pose combination consists of at least two poses to enable triangulation, which means that there are

$N+1$ combinations that are not valid: the combination having no poses and N combination having only one pose. Although $\left(\prod_{j=1}^{N}(n_j+1)\right) - (N+1)$ is not necessarily a large number, we can avoid calculating invalid combinations by representing the problem as a graph and using graph theory to navigate the state space efficiently. The goal is to find the combinations having the most corresponding poses. The Hungarian algorithm may come to mind, but we do not know the number of people in the scene beforehand. Only the minimum and the maximum number of people can be determined based on the observations from each camera. Therefore, another approach is used.

As a first step towards the triangulation of multiple poses in a scene, pairwise correspondences are evaluated. A fully connected weighted graph is constructed where each vertex represents a 2D pose. The connections between vertices have a weight equal to the maximum reprojection error of the points in the pose after pairwise triangulation (see section 6.3). However, to reduce computation time and avoid keypoint ambiguity issues (see later), the evaluation is limited to two keypoints: the midhip and the neck. Invalid connections are assigned a weight of ∞. That includes connections between poses from the same view and connection for which the lines do not intersect within the region of interest. Typically, the reprojection errors are small when the points are accurately detected and corresponding to the same 3D point. A threshold is used to prune the connections with considerable weight. The initial weights are discarded from the pruned graph.

6.4.2 Clustering and triangulation

The new graph is no longer fully connected and dramatically reduces the state space compared to the original problem. The pose combinations with the most views involved manifest as maximal cliques in this new graph. These are the combinations that need to be found. The search for cliques is NP-complete, but the clique size is limited. The brute force search algorithm has running time $\mathcal{O}(n^k k^2)$, where n the number of vertices is, and k is the size of the clique. The good news is that k is equal to the number of cameras N, which is significantly lower than the number of vertices in the graph.

At first, the maximal cliques of size N are searched. Once a valid combination is chosen, the poses involved in that combination cannot be part of another combination and, therefore, these connections are pruned. It means that the number of edges between poses is further

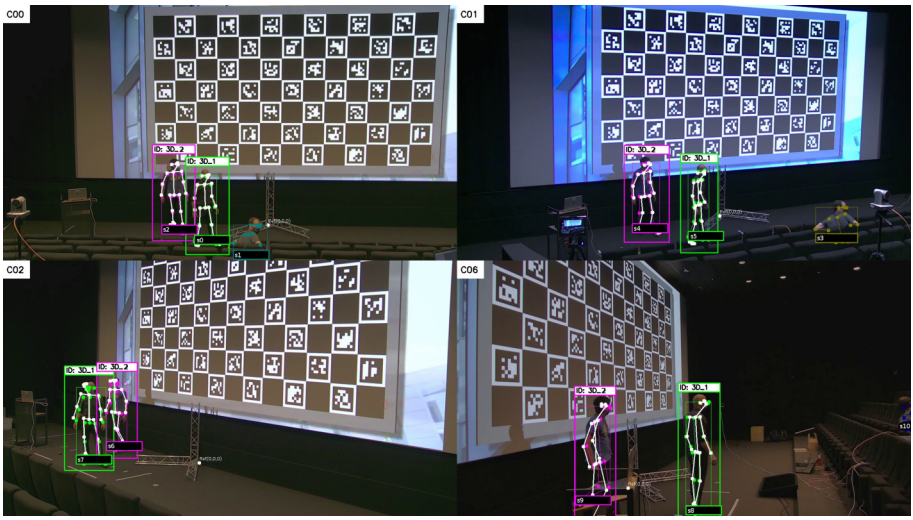


Figure 6.11: *Result of the cross-view pose correspondence in a camera setup using four cameras. The two people on stage are correctly matched in the different views. Note that the views of C00, C01 and C06 actually contain a third person, however, this person is not reconstructed because he sits outside our region of interest.*

reduced after each iteration, and therefore the state space becomes smaller and smaller. The cost of a maximal clique in the graph is defined as the maximum triangulation of the midhip and neck keypoint using all poses part of the maximal clique to guard the cycle consistency constraint. Cycle inconsistency can be observed when the cost of such a clique is significantly higher than the individual costs of the pairwise matches in the clique (above the threshold used for pruning the weighted graph).

6.4.3 An example

Let us consider the example in Figure 6.11. We see four cameras in which the number of detected poses is 3, 3, 2 and 3, respectively. The goal is to find the cross-view pose correspondences (the pink poses and the green poses). The poses are represented in a graph in Figure 6.12. The graph has a significant number of connections, mainly since the two persons on stage are standing close together and, therefore, only a few edges can be pruned based on a threshold of 50 px. A smaller threshold may result in pruning valid edges. The value of 50 px is rather large and should not be exceeded if two corresponding poses are triangulated unless the camera calibration is of really poor quality.

In this graph, the maximal cliques are searched. A total of 16 different maximal cliques can be found. Clearly, many of these maximal cliques are in conflict with each other because they do not form a collection of mutually disjoint sets. In order to find the valid combinations, we calculate the cost for each maximal clique. The results are ranked in ascending order in Figure 6.14c. Note that many maximal cliques have a cost higher than 50 px, which indicates that these connections are cycle inconsistent. Non-corresponding poses are combined in these cliques. These connections are crossed out in Figure 6.14c.

The particular example shows that the maximal clique $[s_2, s_4, s_6, s_9]$ has the lowest cost. The algorithm chooses this clique and prunes all other edges connected to s_2, s_4, s_6 or s_9 because these poses can no longer be part of a combination with other poses. Figure 6.14a shows the pruned graph, where $[s_2, s_4, s_6, s_9]$ is indicated in green. The maximal clique $[s_0, s_5, s_7, s_8]$ remains valid and is therefore also accepted. There are no maximal cliques of 4 left in Figure 6.14b. There is, however, a maximal clique of three: $[s_1, s_3, s_{10}]$. This maximal clique corresponds to the person in the audience. For the sake of this example, no region of interest was defined to illustrate that maximal cliques of valid poses are not necessarily of equal size. Imposing a ROI mitigates intersections outside this ROI by assigning a weight of ∞ to the corresponding connections.

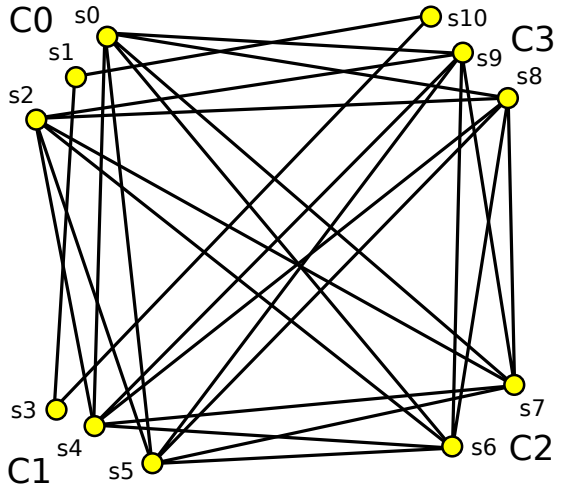
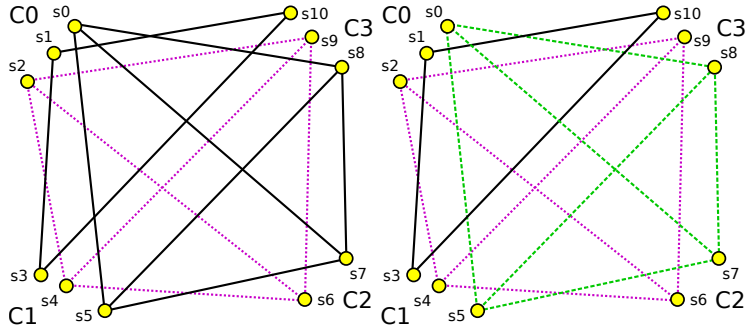


Figure 6.12: *Graph that represents all possible matches between poses from different views after pruning the connection where the reprojection error is too large. In this graph the maximal cliques will be searched to find the correct cross-view pose correspondences.*



(a) *Iteration 1* ***(b)*** *Iteration 2*

Clique	Cost	Clique	Cost	Clique	Cost
$[s_2, s_4, s_6, s_9]$	5.78	~~$[s_2, s_5, s_6, s_8]$~~	~~70.14~~	~~$[s_0, s_5, s_6, s_9]$~~	~~104.30~~
$[s_0, s_5, s_7, s_8]$	12.40	~~$[s_0, s_4, s_7, s_9]$~~	~~75.89~~	~~$[s_2, s_4, s_6, s_8]$~~	~~124.62~~
$[s_0, s_5, s_6, s_8]$	37.12	~~$[s_2, s_4, s_7, s_8]$~~	~~80.57~~	~~$[s_2, s_5, s_7, s_9]$~~	~~137.97~~
$[s_2, s_4, s_7, s_9]$	46.91	~~$[s_0, s_4, s_7, s_8]$~~	~~89.07~~	~~$[s_0, s_5, s_7, s_9]$~~	~~147.13~~
~~$[s_2, s_5, s_7, s_8]$~~	~~53.56~~	~~$[s_2, s_5, s_6, s_9]$~~	~~91.96~~		
~~$[s_0, s_4, s_6, s_9]$~~	~~57.46~~	~~$[s_0, s_4, s_6, s_8]$~~	~~103.52~~		

(c) *Clique costs*

Figure 6.14: *Pruning the graph. Maximal cliques are coloured and shown in dashed lines. Combinations are ignored if the maximum reprojection error exceeds 50 pixels.*

6.5 3D keypoint reconstruction

3D point reconstructions have been discussed in detail in Section 6.3 where we assume that points are correctly labelled. However, some difficulties may arise in practice, specifically when a pose estimator estimates the 2D points. Points may be mislabelled, their location may be poorly corresponding to their actual location, or some may not be detected.

A pose estimator, such as OpenPose, provides a confidence score in the range 0.0-1.0 for each of the detected keypoints. In general, the positions of the keypoints are relatively accurate. However, when this confidence score of a point is low (e.g., below 0.2), it is better to discard these points during triangulation because their position is inaccurate and unreliable. Another issue with the pose estimator is confusion between the left and right side of a person's body. Such confusions lead to mislabelled keypoints which has a big impact on the triangulation process. In the following sections, we discuss the errors in more detail.

6.5.1 Occluded joints

Pose estimators may not detect all keypoints due to occlusion, or worse, detect occluded keypoints in the wrong place. Figure 6.15 shows a recording from the Xiak dataset where the operator performs tasks at a conveyor belt. The operator is partly occluded in three camera views: C1, C3 and C4. We notice that OpenPose reports positions for some of the occluded keypoints, such as the knees. These detections are often not in the correct position and may cause issues in the reconstruction process. However, OpenPose generates a confidence score for each keypoint and that score is typically low for occluded keypoints (Table 6.2). The confidence scores for the left and right knee keypoint in each view are shown in the table below the figure. In general, the occluded views report low scores (≤ 0.55) while the other views report higher scores (≥ 0.7). It is safe to use a threshold of 0.3 in this dataset.

We see that missing keypoint detections and erroneous keypoint detections do not immediately cause inaccuracies in our proposed 3D human pose estimation framework. Each keypoint is reconstructed independently which means that the number of cameras can vary between two and N. The 2D keypoints with a low confidence score are not considered for triangulation. Note that at least two cameras need to observe each keypoint to triangulate its position. This example illustrates that the proposed system effectively handles occlusion in these situations. In

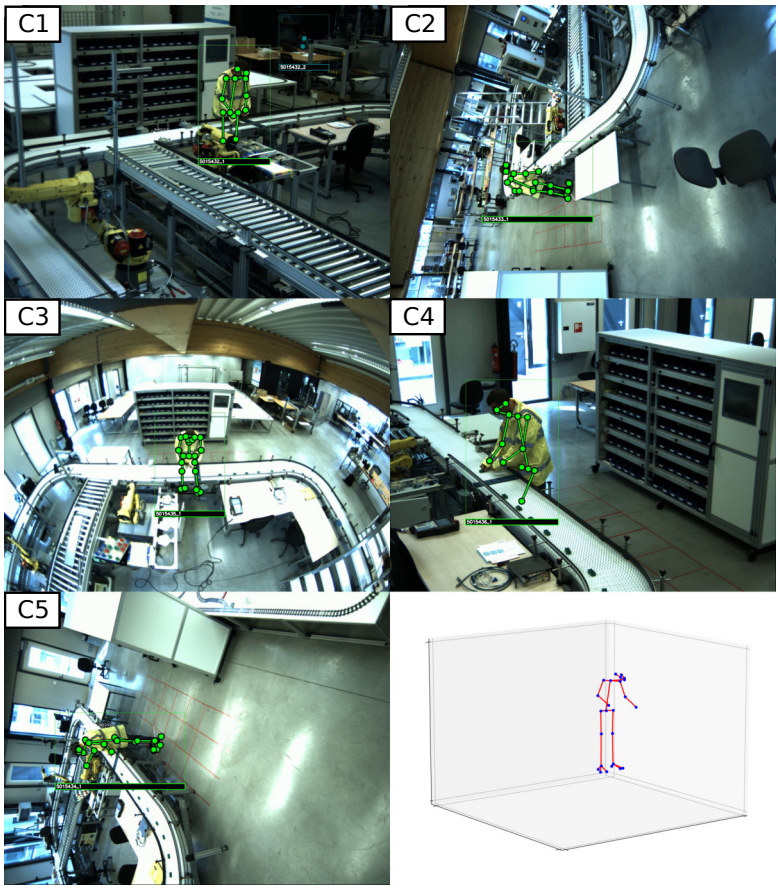


Figure 6.15: *In the Xiak dataset, a number of keypoints, such as the knees are frequently detected in the 2D poses, despite being occluded in camera C1, C3 and C4. However, the 3D reconstruction is still triangulated correctly. Table 6.2 shows the confidence scores of the right and left knee for each view.*

Section 7.4 we will show that we can combine the occlusion detection from Chapter 5 to effectively remove the occluded keypoints, which is more reliable than relying on a fixed confidence score threshold.

6.5.2 Typical pose estimation errors

In the specific case of multi-view 3D pose estimation, there are several typical errors. On the one hand the pose estimator may introduce errors: limb switch (I.), double limb (II.) and misdetected limb (III.). On the other hand, the camera infrastructure may also cause errors in frame drop (missing frames). We will discuss each error in detail and hint

Keypoint	C1	C2	C3	C4	C5
RKnee	0.18	0.84	0.49	0.00	0.70
LKnee	0.21	0.88	0.53	0.06	0.86

Table 6.2: *Confidence scores of the occluded and visible keypoints in Figure 6.15. The occluded keypoints in camera C1, C4 and C5 show low confidence scores for the occluded knee keypoints and higher confidence scores for visible keypoints.*

towards a solution. In section 6.5.3 we will analyse the frequency of each error in one of our datasets to indicate their importance.

I. Limb switch error

A limb switch error means that the identities of the left leg and the right leg are switched. All the keypoints belonging to the left leg are assigned to the corresponding keypoints on the right leg and vice versa. OpenPose uses a frame by frame approach. It means that legs can be switched from frame to frame. Limb switch errors often occur on a lateral view of a person because the perspective transformation creates ambiguity between the right and left leg. E.g., from a lateral view of a running athlete, it is sometimes hard to tell which leg is in front and which leg is in the back. Even for humans, it is hard to tell based on a single frame in these cases. Figure 6.16 illustrates this issue in 3 consecutive frames where the second image confuses between the left and right leg. Temporal 2D tracking of all keypoints may solve this problem. In that case, swapping the left with the right leg measurements may fill in the gaps in a track. However, we need to be careful not to solely rely on 2D tracking because limb switching may track the left and right side incorrectly for an extensive period. Therefore, we decided to solve the limb switching problem in the triangulation process by testing different hypotheses for limb switches from different views, for which there are hints that the limb switch errors have occurred.

II. Double limb error

We define a double limb error as the error occurring when keypoints for both the left and right leg are detected, but all are detected on the same leg in the image, even though the other leg is also visible. Also, this error occurs most often on a lateral view of a person. The misdetected leg needs to be ignored in the reconstruction, and the predicted location can be used instead.

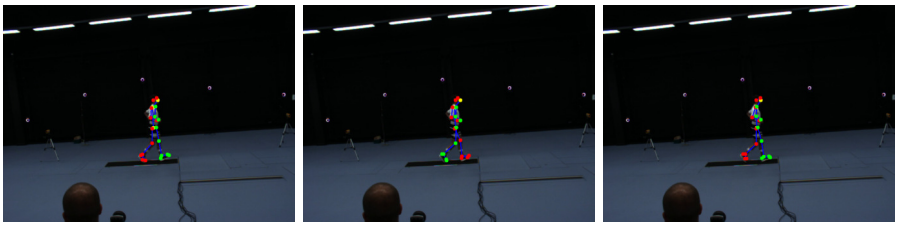


Figure 6.16: *Example of a limb switch error. 3 consecutive frames are displayed where the left (green) leg is swapped with the right (red) leg.*

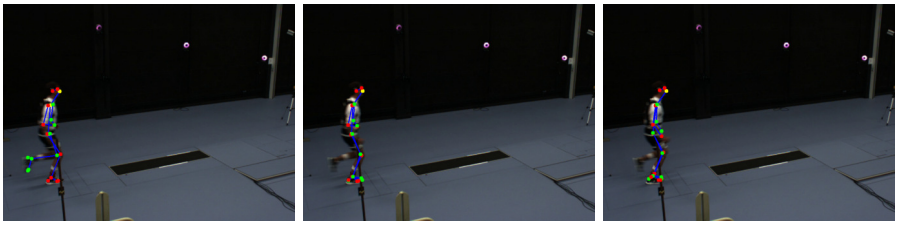


Figure 6.17: *Example of double limbs. 3 consecutive frames are displayed where the second and third frame showcase the issue.*

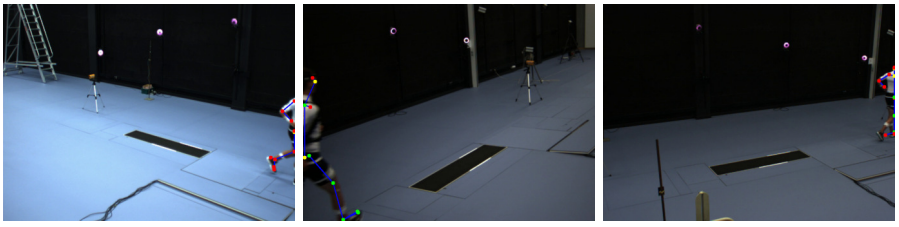


Figure 6.18: *Examples of misdetected limbs. These errors often happen on the image border where a person is only partially visible.*

III. Misdetected limb error

The third type of pose estimator errors is a container for all the other detection-related errors. In general, these errors occur when a person is only partially visible, e.g., on the image border or when a person is partially occluded. Figure 6.18 shows three examples of misdetected limbs: the left leg is mistaken for the right or vice versa when only one leg is in view (first and second image) or the leg detected is completely wrong (third image). These errors are hard to cope with because their positions may be unreliable as well. Such errors are usually solved by ignoring outliers in the triangulation process.

IV. Frame drop

The fourth type of errors is not related to pose estimators but to the camera infrastructure itself. However, this type of error also has an impact on the reconstruction result. We often use cameras with ethernet connection to a processing pc or action cameras such as GoPros in places where wires cannot easily be installed. Unfortunately we have to cope with frame drops during the recording process, either due to network issues of a receiver that is not fast enough or not configured properly. The camera setup in Sport Science Lab Jacques Rogge can record 4 simultaneous video streams at 67 fps in each of the 2 capture PCs. The cameras produce bayer images of 780x580 pixels (1 byte per pixel) which are sent uncompressed of the network. Therefore, the network is almost saturated as 0.97 Gbps of the available 1 Gbps is required. If a frame is dropped over TCP there is basically no room to resend the frame and it will arrive too late to store in the video container.

It is crucial to keep the different video streams synchronised while processing the data. Therefore, the recording process needs to guarantee that dropped frames will be handled correctly, either by inserting frames to replace missing data or by logging timestamps for every received frame. For instance, a GoPro camera adds empty frames in the video when a frame was dropped. Empty frame detection or timestamps can help to process a dataset without losing synchronisation. When a frame is missing, we can either predict the pose in the missing frame based on previous and next poses, or ignore that particular view completely.

6.5.3 Error frequency

We analysed part of the SSL-JR dataset (Figure 6.19). 13 sequences were considered out of 33, which means 1318 synchronised frame sets from 7 cameras. Figure 6.20 shows the different kinds of errors in that part of the dataset. Besides the keypoint detection errors, we also investigated the amount of frame drop per camera. While annotating the different errors, we noticed that several errors occur on the image border when the person of interest is only partially visible. Roughly 50% of all pose-related errors occur on the image border.

There is a significant imbalance between the errors produced by each camera. We can explain this using a layout of the camera setup of this particular dataset. The arrow in the camera setup in Figure 6.19 indicates the track of the running person in this dataset. Camera C4, C5 and C7 have the best rear and frontal views of the person. Therefore,

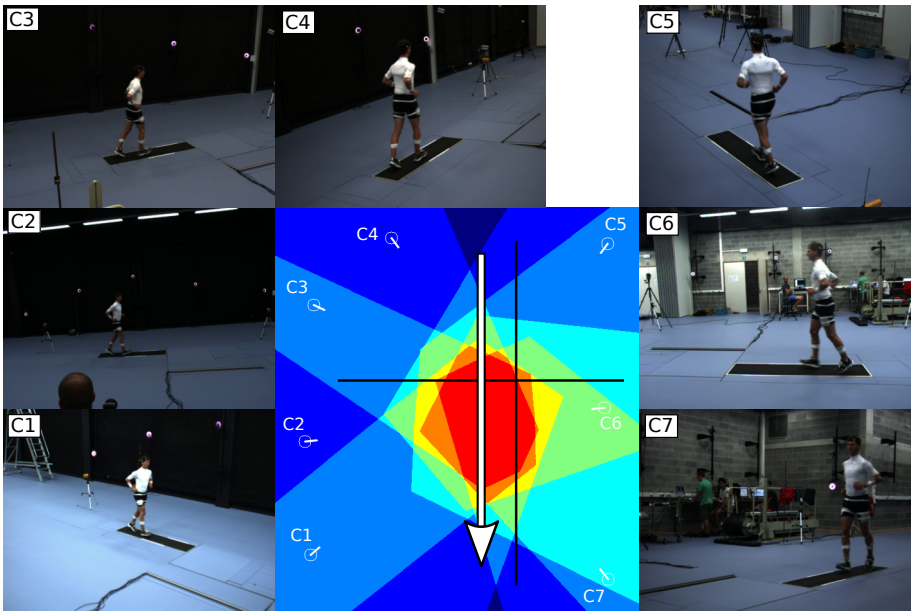


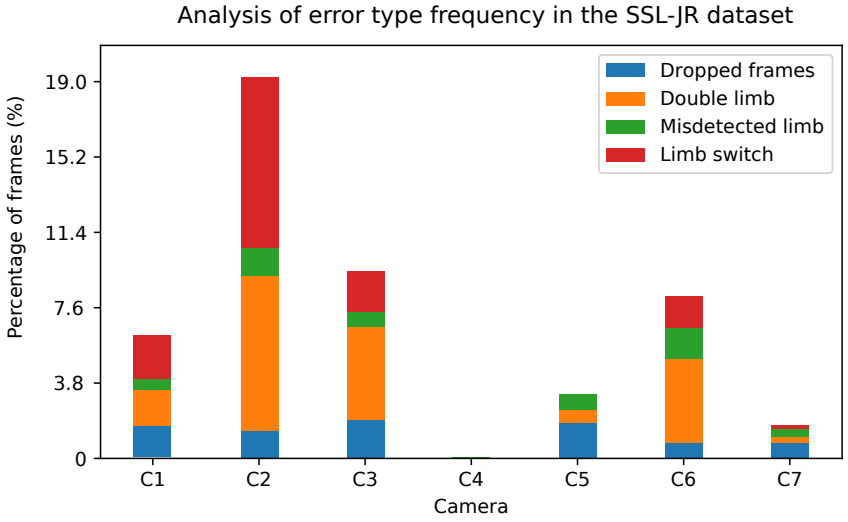
Figure 6.19: *Dataset of a runner recorded in the Sports Science Lab Jacques Rogge (SSL-JR) in Ghent.*

they have fewer issues with ambiguous limbs. Cameras C1, C2, C3 and C6 capture the subject more from a lateral perspective where ambiguous detections are common. Primarily camera C2 produces many errors because the camera's viewing direction is perpendicular to the running direction. The camera is also further away from the person of interest compared to camera C6. Note that the camera coverage map also shows that out-of-view errors will need to be handled because the camera coverage is not the same for all positions of the running person.

If only one camera produces an error, the erroneous keypoints can be simply treated as outliers and can be ignored from the calculation. However, we noticed that errors might occur in different views at the same time. A particular hard case to handle is when multiple cameras confuse the left and right limbs. A valid 3D reconstruction can be calculated, but the labels of these limbs may be switched in the reconstruction for that particular frameset. We analysed how often an error occurs in different camera views simultaneously. We define a frameset as the set of frames captured simultaneously from different camera views. Figure 6.21 shows the fraction of erroneous framesets: 38.09% frame sets experience an error in a single view, 4.93% of framesets experience errors from two views, 0.23% of framesets experienced errors in three views (all frame drop issues). The total fraction of framesets that experience any error

Camera	dropped frames	double limb	misdetected limb	limb switch
C1	1.593%	1.821%	0.531%	2.200%
C2	1.366%	7.815%	1.366%	9.256%
C3	1.897%	4.704%	0.759%	2.049%
C4	0.000%	0.000%	0.076%	0.000%
C5	1.745%	0.683%	0.759%	0.000%
C6	0.759%	4.259%	1.517%	1.593%
C7	0.759%	0.303%	0.379%	0.152%

(a) *Error type frequency*



(b) *Bar graph error type frequency and camera setup*

Figure 6.20: *Analysis of error type frequency in the SSL-JR dataset. The errors are view-dependent (see camera setup). Double limb errors and limb switch errors occur most often.*

is 43.25%, which indicates that these errors are significant. Therefore, we should deal with them. Fortunately, the majority of these frame set errors are single frame errors.

The errors are not entirely independent. The errors related to OpenPose depend on the relative position of the athlete and the camera since lateral views of the person increase ambiguity and double limb and misdetected limb errors often occur when the athlete is captured near the image border. Frame drop errors on the other hand seem more independent at first but performing some joint probability calculation based on the

table in Figure 6.20a, we notice something different. Let us define p_{fd} as the independent probability that frame drop occurs in a camera. To calculate the joint probability of a frame drop simultaneously occurring in exactly k of N cameras, we can use the formula:

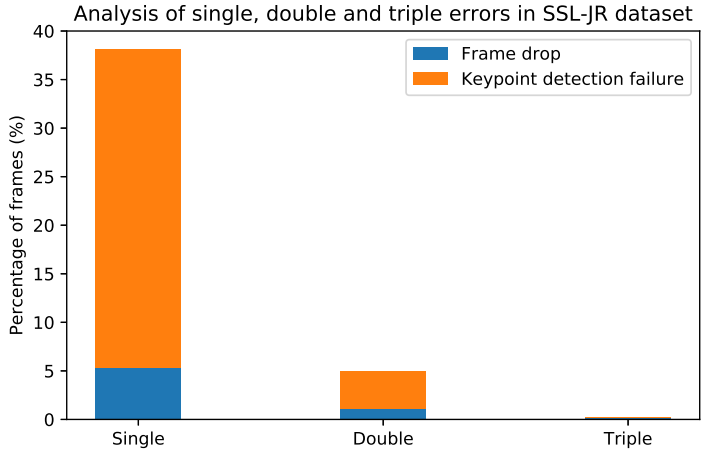
$$\binom{N}{k}(p_{fd})^k(1-p_{fd})^{N-k}. \tag{6.4}$$

With an average probability p_{fd} of 1.16% (see table in Figure 6.20a), we calculate the expected frame drop for single, double and triple views as 7.6%, 0.27% and 0.0052% respectively. Comparison with Figure 6.21 shows that the number of single frame drop errors is smaller than expected (5.31%), while double and triple frame drop errors are larger (1.06% and 0.23%). It means that single frame drop errors occur significantly less than expected while double and triple errors occur more often than expected. This indicates that the events are not entirely independent. The reason is most likely that the recording tool is unable to keep up with all the images being transferred over the network. In that case it is more likely that the recordings fails for multiple cameras at once.

The number of erroneous framesets is large because seven camera views are involved in this dataset, each having a specific chance of producing an error. This does not mean that a large set of cameras leads to a bad 3D reconstruction, but means that a reconstruction algorithm needs to be aware of these errors and provide a suitable solution to handle them appropriately.

6.5.4 Handling limb ambiguities

The data analysis in the previous section helped to develop a strategy to handle these errors. On the one hand, we handle frame drops to avoid using irrelevant measurements. On the other hand, we assess the reliability of each detected pose based on several simple tests. We ignore poses with too few keypoints (these often occur on the image border or do not even correspond to a person, e.g., a camera tripod that is recognised as a person). We also validate each keypoint of the keypoint model in each view to detect sudden position changes using a simple spatio-temporarily tracker. In the case of limb switching, the previous left leg keypoints are closer to the current right leg keypoints and vice versa. In case of a double limb error, the leg keypoints of one leg are wrongly detected, which can be noticed by a 2D tracker. These error detections



error types	misdetection	frame drop	total
single view errors	32.78%	5.31%	38.09%
double view errors	3.87%	1.06%	4.93%
triple view errors	0.00%	0.23%	0.23%
			43.25%

Figure 6.21: *Analysis of simultaneous error frequency in different camera views. Single frame errors are common (about 38% of all frames), double frame errors still account for 5%. Triple frame errors or more are uncommon.*

feed a decision mechanism that tests some hypotheses (combinations of adapted poses). The number of hypotheses to test may be large when many cameras make errors simultaneously, but from Figure 6.21, we see that simultaneous errors are rare. There was no noticeable drop in computation speed in the performed experiments.

To clarify the mechanism, we give an example with four cameras where C1 has a limb switch error when C3 has a double limb error. Let us indicate S_i as the detected pose, $S_{i,LS}$ as the pose corresponding to S_i where the left and right leg keypoints are swapped and $S_{i,DLl}$,$S_{i,DLr}$ as the poses corresponding to S_i where the left and right leg keypoints are removed respectively. When the 2D tracker successfully detects the occurrence of these certain types of errors, six combinations will be tested. The pose with the lowest maximum reprojection error will be chosen as the current valid 3D pose (Table 6.3).

#	C_1	C_2	C_3	C_4
1	S_1	S_2	S_3	S_4
2	$S_{1,LS}$	S_2	S_3	S_4
3	S_1	S_2	$S_{3,DLl}$	S_4
4	S_1	S_2	$S_{3,DLr}$	S_4
5	$S_{1,LS}$	S_2	$S_{3,DLl}$	S_4
6	$S_{1,LS}$	S_2	$S_{3,DLr}$	S_4

Table 6.3: *Possible pose combinations in case of limb switch (C1) and double limb error (C3). These combinations (hypotheses) are tested and the pose with the lowest maximum reprojection error is selected.*

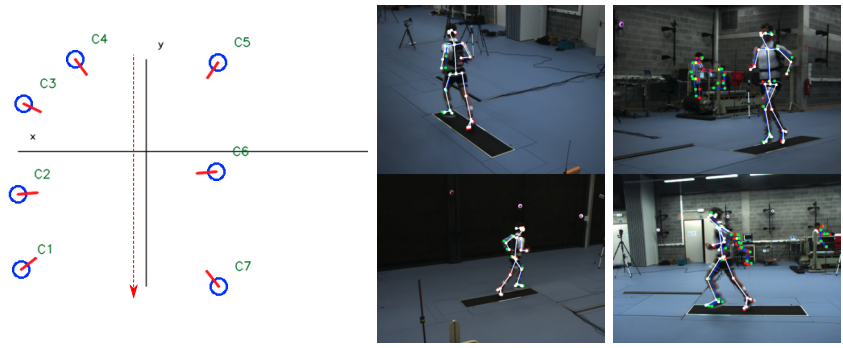


Figure 6.22: *Camera setup in the Sport Science Lab Jacques Rogge in Ghent on the left with a red arrow indicating the running direction (SSL-JR dataset). On the right we show four camera images with the estimated 3D pose (white).*

6.6 Experimental results

Two experiments were conducted on two different locations which both were recorded with RGB cameras and infrared cameras for generating the ground truth. The marker-based camera systems (Qualisys and Vicon) have a theoretical submillimetre precision for the marker positions. However, we should keep in mind that due to marker/soft tissue movement, it is unlikely that submillimetre precision is reached to calculate keypoint centre positions. Either way, these maker-based systems are often used to measure keypoint positions in biomechanical research. However, for many applications, submillimetre precision is not needed. A precision of 1-2 cm is accurate enough to assess a person's macro-movement.

The first dataset was recorded at the Sports Science Laboratory Jacques Rogge (SSL-JR) at Ghent University. The vision-based system consisted

of seven 4.5 MP cameras (Manta G-046C, AVT, Stadtroda, Germany). A person ran in a straight line, always in the same direction at different speeds ranging from 2.1 to 5.1 m/s (Figure 6.22). The camera images are captured synchronously by two computers at 67 Hz. The running length that can be captured is around 11 meters. The infrared-based motion capture system consisted of ten 1.3 MP cameras (Oqus3+, Qualisys AB, Göteborg, Sweden) operating at a frame rate of 250 Hz. The cameras were fixed to the lab walls, uniformly distributed to measure 4 m of the running length, with a distance to the centre of the volume ranging from 3.5 to 7 m. In total, 88 Passive IR-reflective, 12 mm-sized spherical markers were attached to the subject body and used for complete body modelling in Visual 3D software (C-Motion Inc., Germantown, USA). Joint centre coordinates of the ankles, knees and hips were exported for comparison. The wand calibration of the setup showed a standard deviation on measured distances of 0.4 mm.

The second dataset was recorded in Leuven, Belgium. Only three 4.5 MP cameras (Manta G-046C, AVT, Stadtroda, Germany) were used operating at 50 Hz. The cameras were located closer to the person of interest than the first dataset because the measuring volume was only 3 x 3 x 2 meters. The person in this dataset is executing stationary movements such as squats and clocks (Figure 6.23). A fixed ten camera Vicon system (Vicon MX T20, VICON Motion Systems Ltd., Oxford, UK) supplemented with three additional portable Vicon Vero cameras (Vicon Vero v1.3, VICON Motion Systems Ltd., Oxford, UK) were used. All cameras were sampled at 100 Hz and had a measurement error of 1 mm. In addition, ground reaction forces were collected using two AMTI OR 6 Series force plates sampled at 1000 Hz (Optima, Advanced Mechanical Technology, Inc., Watertown, USA). These force plates were used to determine initial ground contact during the side cut manoeuvre and check the execution of the clock for the Vicon system. A researcher placed 39 retro-reflective markers on the participant using palpation to identify the correct attachment site. Markers were placed as shown in Figure 6.23.

For the triangulation we used ℓ_∞-norm optimization, but the difference with the ℓ_2 norm is limited because many outliers are already removed by using the 2D trackers. We filtered the raw measurements in the spatio-temporal domain with a Hanning window, of length 7 (0.1 seconds). For offline processing, 3 frames from the past and 3 frames from the future are used. This filtering slightly improves the results. Figure 6.24 shows a typical graph produced by the proposed system after smoothing. We see that the proposed system follows the marker-based positions rather

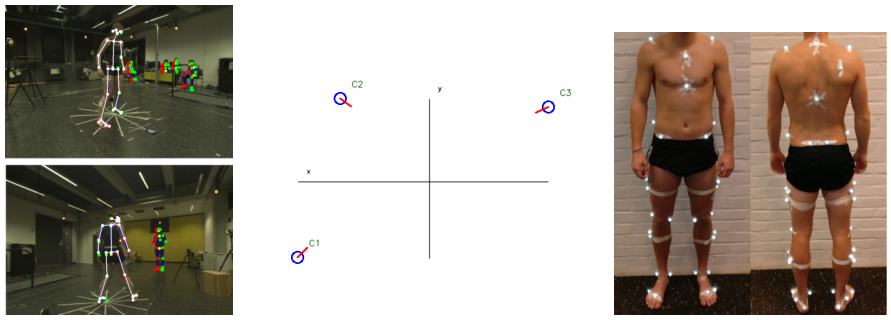


Figure 6.23: *In the camera setup in Leuven, only three cameras were used. However, the cameras are positioned much closer to the test subjects, approximately 2.5 m. Therefore, a more accurate camera calibration was possible.*

accurately. We will evaluate results based on average errors and their standard deviation.

Table 6.4 shows the accuracy averaged per dataset, while Figure 6.25 shows the distribution of these numbers. We may conclude that spatio-temporal filtering improves the results by decreasing the standard deviation and average positional error between 1 and 3 mm. For the second dataset we notice an offset in positional errors for some joints. The limited number of cameras and their relative position is most likely the cause of the offset. Also, the cameras in this setup are not entirely evenly distributed around the person of interest.

The accuracy of the keypoint positions depends on the accuracy of the camera calibration and the pose estimator. Therefore we conducted an extra experiment to figure out the impact of the camera calibration. We measured eight ground truth positions in the reconstruction volume and determined their corresponding pixel coordinates. Using these pixel coordinates, we triangulated the points to obtain 3D points and compare them with the ground truth positions. We also measured the reprojection error (the difference between the pixel coordinates and the projection of the triangulated point). These results are shown in Table 6.5. The average 3D triangulation error in the SSL-JR dataset is 21.0 $\pm$ 5.8 and 11.3 $\pm$ 4.7 in the IPLAY-Leuven dataset. It means that the calibration errors account for about half of the positional errors. The remaining error is the actual error from the triangulation of the points detected by the pose estimator. The SSL-JR dataset error is larger than the error in the IPLAY-Leuven dataset because the cameras are further away from the test subject. Small angular calibration errors correspond to larger errors further away from the camera, which is why

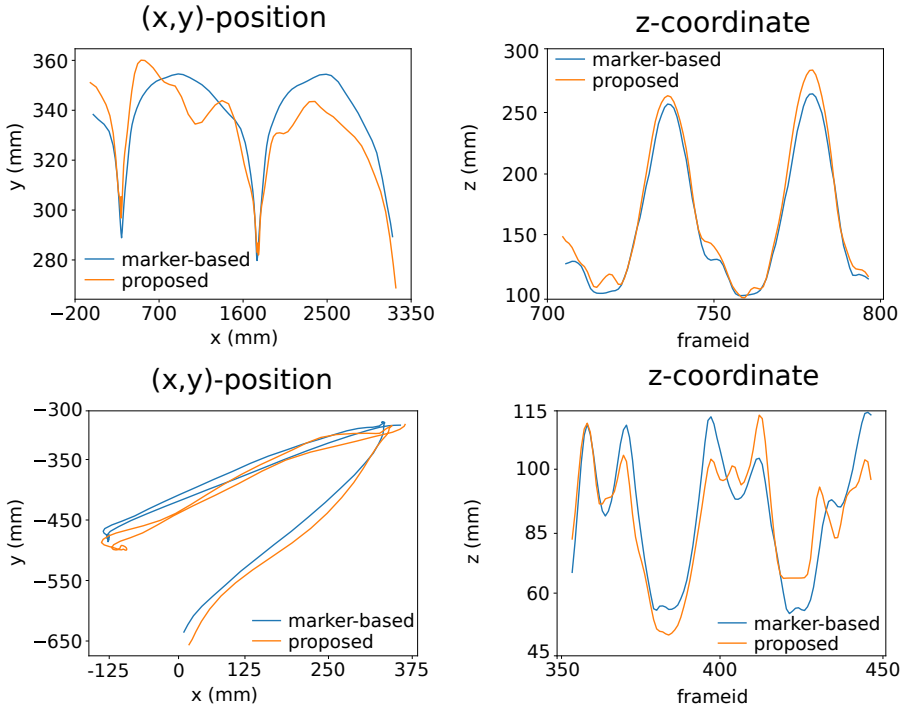


Figure 6.24: *Typical result of the positions of an ankle keypoint (top row: SSL-JR dataset, bottom row IPLAY-Leuven dataset). Note the different scales in each of the graphs.*

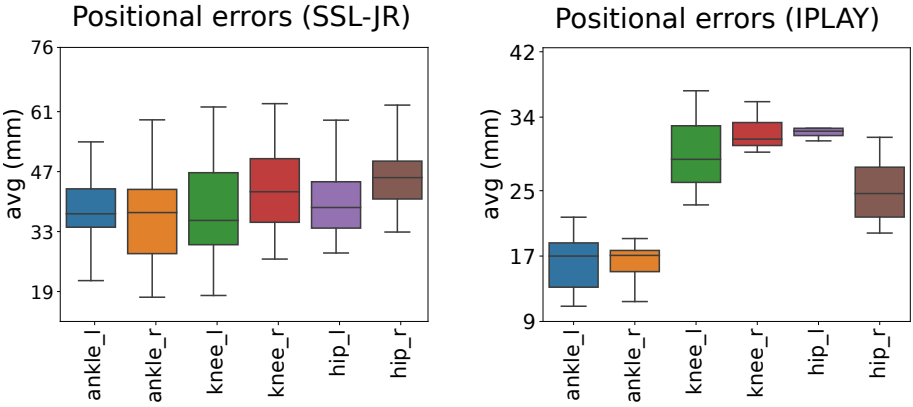


Figure 6.25: *Average positional error between the marker-based positions and the estimated 3D position for all 33 sequences (SSL-JR) and 9 sequences (IPLAY).*

		Unfiltered		Filtered	
	keypoint	**avg**	**stddev**	**avg**	**stddev**
S	ankle_l	42.1	23.7	40.6	21.3
S	ankle_r	41.2	21.7	38.8	19.4
L	knee_l	42.2	20.8	40.4	19.1
J	knee_r	46.3	19.9	44.8	18.3
R	hip_l	44.4	18.8	41.6	15.5
	hip_r	50.7	15.2	49.2	12.5
I	ankle_l	19.4	11.9	18.7	8.2
P	ankle_r	16.7	9.6	15.9	6.6
L	knee_l	30.3	10.6	29.8	7.9
A	knee_r	32.6	12.4	32.1	10.0
Y	hip_l	34.2	15.7	33.5	10.9
	hip_r	26.1	14.7	25.1	11.0

Table 6.4: *Positional errors and standard deviation averaged over 33 sequences (SSL-JR dataset) and 9 sequences (IPLAY-Leuven dataset). All measurements are in mm.*

	SSL-JR	IPLAY
Reprojection error (px)	2.2 ± 0.5	1.2 ± 0.1
3D triangulation error (mm)	21.0 ± 5.8	11.3 ± 4.7

Table 6.5: *Analysis of the accuracy of the camera calibration expressed in reprojection error and triangulation error. These errors are significant and account for about half of the errors in Table 6.4.*

the positional errors are smaller in the IPLAY-Leuven dataset. Because the subject is larger in the image, the pose estimator can detect the keypoints more accurately.

6.7 Conclusion

This chapter presented a fast and reliable method to convert 2D poses from multiple camera views into 3D poses. The proposed method can handle occlusion and can reconstruct multiple poses from different people in the scene. The proposed method copes with misdetected keypoints and confusion between the left and right part of the person to extract reliable 3D tracking data for 25 keypoints of the human body (OpenPose BODY_25 model). During our experiments, we found that the positional error for the lower limbs is between 15.9 and 49.2 mm and the standard deviation between 6.6 and 21.3 mm. We compared our system to marker-based systems, which report submillimetre accuracy. The accuracy of our method is not as precise as that of a marker-based system with the used camera setup. However, our method is has three strong benefits compared to marker-based systems. First, our solution is much more flexible because it can be used in larger areas e.g., an athletics track. Second, in contrast to a marker-based system, our solution also works outdoors. Third, if pose estimators become more reliable, the accuracy of our system will also increase.

The results from the setup in Leuven show that lower errors are possible using cameras closer to the subject. The reason is twofold. On the one hand, persons appear more prominent in the images, which means more accurate detections by the pose estimator. On the other hand, the cameras can be calibrated more accurately because the point correspondences in the different cameras are also more accurate. More of these cameras would, therefore, generate more accurate results. Also, cameras producing images with higher resolution can increase accuracy for the same reasons. With these improvements, athlete techniques can be accurately measured.

Other typical application can be found in the entertainment industry. Due to feedback from the system to the player, the player will react on less accurate positions by moving a bit more to reach the goal location in the game. Macro-movements should be detected correctly, and that is no problem for the proposed method. Applications, where a behaviour classification is more important than the exact pose are also suitable candidates for the proposed system. Such applications are, for instance, presenter analysis to record a presentation automatically. It is essential to know where the people are and what they are doing in these applications, but a millimetre-precision pose estimation is usually unnecessary. Besides, the technology should often be non-intrusive, so markerless systems are often preferred.

The main contributions made in this chapter are the following:

- reliable cross-view pose correspondence based on pairwise triangulation and maximal clique searching in a graph;
- handling limb ambiguities by correcting the 2D poses and hence produce reliable 3D poses, even with erroneous 2D poses;
- real-time 3D pose estimation using OpenPose pose estimator from multiple cameras;
- handling occluded keypoints in some camera views.

Some of the work in this chapter is described in these publications:

- M. Slembrouck, H. Luong, J. Gerlo, K. Schütte, D. Van Cauwelaert, D. De Clercq, B. Vanwanseele, P. Veelaert, and W. Philips. Multiview 3d markerless human pose estimation from OpenPose skeletons. In *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS'20)*, volume 12002, pages 166–178. Springer, 2020. ISBN 9783030406042
- B. B. Nyan, M. Slembrouck, P. Veelaert, and W. Philips. Distributed multi-class road user tracking in multi-camera network for smart traffic applications. In *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS'20)*, volume 12002, pages 517–528. Springer, 2020. ISBN 9783030406042

Chapter 7

Projects and Applications

The work in this book has been applied to several use cases. Volumetric shape reconstruction and 3D pose estimation are valuable ways to understand what is happening in the scene. While volumetric shape reconstruction defines the occupied space of an object, 3D pose estimation offers a more detailed analysis of the different body parts of a person. It can be used as input for behaviour analysis. Note that both the volumetric shape reconstruction and the pose estimation are usually only a part of a larger system. Therefore, it can be considered as middleware for numerous applications.

Table 7.1 shows several applications of our proposed solutions. For each application, we indicated which parts of this work were important. Some applications avoid occlusions, such as gait analysis and game control. Some applications do not necessarily require shape reconstruction, such as game controlling and expert recordings. The applications centered around assembly line analysis benefit from combining all three techniques. Our methods perform best when the cameras are accurately calibrated. Therefore, we will focus on applications where we can obtain reasonable accuracy, typically in reconstruction volumes of up to 10 m $\times$ 10 m $\times$ 3 m.

In this chapter, we will discuss some of these diverse applications and their challenges. Each application requires a specific solution, and we will explain how we meet the requirements. In the first part of this chapter, we will discuss applications that require object reconstruction and in the second part, applications on person reconstruction.

Application	Section	Shape	3D pose	Occlusion
Gait analysis	7.1	x	x	
Gaming and rehabilitation	7.2		x	
Expert recordings	7.4		x	x
Assembly line analysis	7.4	x	x	x

Table 7.1: *Overview of features used for different use cases in this work.*

7.1 Gait analysis (IRUNMAN)

At Ghent University, the Department of Movement and Sports Sciences conducts research in biomechanical movements of humans. We were involved in the IRUNMAN project where the Sport Science Laboratory - Jacques Rogge was shaped. This lab is used for innovative scientific research in human movement and sports sciences and aims to translate their findings into actual societal applications.

Often such a motion analysis is performed using marker-based systems like Qualisys [81], OptiTrack [67] or Vicon [65]. These systems provide accurate 3D skeletal data of the subjects. Still, they are often impractical to use for tests with many different people because the task of instrumenting a test subject with the necessary markers is time-consuming (up to 30 min per test subject). A full skeletal reconstruction is not required for some research tasks because the analysis may be restricted to a particular part of the body or a macro analysis is sufficient. For instance, a person's step length and running speed can also be extracted from a reliable shape reconstruction.

Therefore, a less intrusive method that offers basic analytical data surely has its merits. We installed a camera network of eight cameras in the Sport Science Lab Jacques Rogge to research which parameters of human motion analysis can be extracted accurately from a shape reconstruction. Each camera captures images with a resolution of 780x580 px at a maximum framerate of 67 fps. The cameras observe an 8 m running area and are calibrated both intrinsically and extrinsically.

In this particular dataset, we investigate if we can estimate typical parameters that describe a person's gait based on the multi-camera system. Therefore the focus is on the lower limbs and, more specifically, on the person's feet. Important parameters to track are cycle times, step lengths and speed.

Figure 7.1 shows the camera coverage map of the camera setup. The arrow indicates the area and direction in which the test subject runs. In

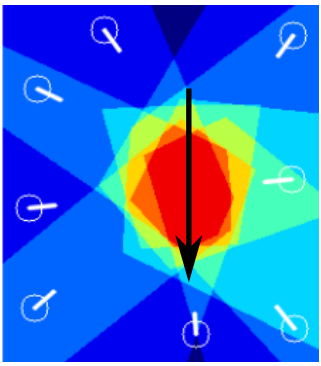


Figure 7.1: *Camera setup (floorplan) and camera coverage map at Sport Science Lab Jacques Rogge. The number of cameras ranges from 0 (dark blue) to 8 (red). This area of maximum camera coverage is limited to focus on a few steps in the gait analysis. The arrow indicates the area and direction in which the test subject runs.*

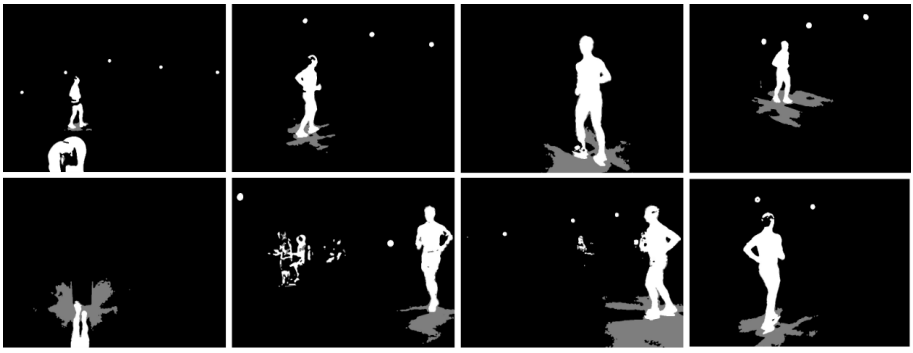
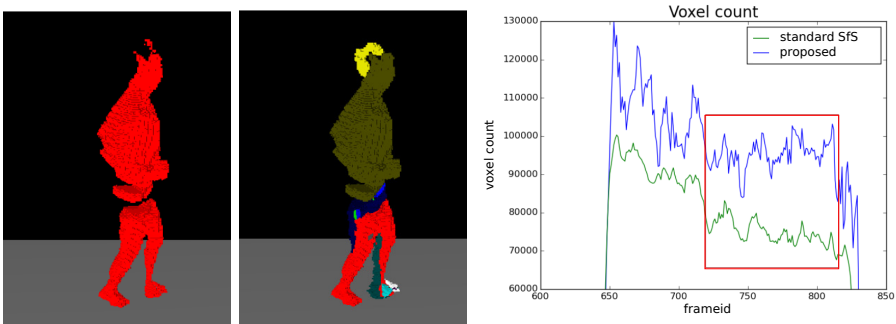


Figure 7.2: *Typical foreground/background segmentation results which are used as silhouettes for the 3D reconstruction.*

the camera coverage map, we see that not all cameras are observing the entire run length. There is a trade-off between the zoom factor (higher zoom factor means more detailed observations, but a reduction of the field of view) and the coverage (larger field of view means that more cameras can observe the same part of the reconstruction volume, but a reduction of details in the observation) of each camera. In this particular use case, we focused on the area covered by all cameras because this is also where the ground truth was captured using the pressure plates and the Qualisys system.

The video streams of the different cameras are captured by two dedicated computers, each handling half of the cameras. The test subject is segmented in each of the camera images to create silhouettes. Since



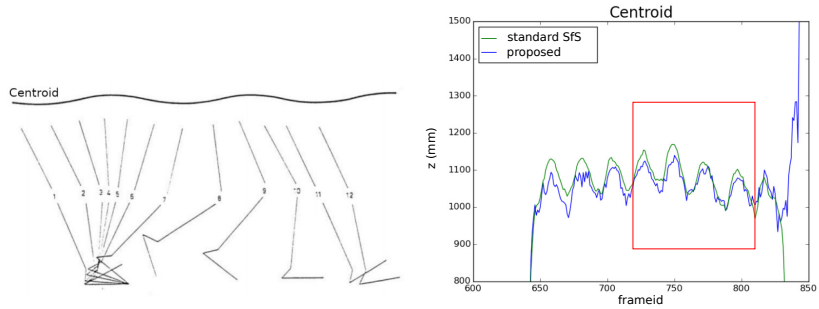
(a) standard SfS *(b) proposed* *(c) voxel count comparison between standard and proposed method*

Figure 7.3: *Results of 3D reconstruction of the standard and the proposed reconstruction method. The rectangle indicates the frames where the test subject is inside the zone of maximum camera coverage.*

the background is predominantly static and the runner is moving, foreground/background segmentation is used. In this particular case, we chose to use the MoG algorithm of Zivkovic [114]. Figure 7.2 shows a set of frames where the runner is segmented. We observe several errors in these silhouettes: missing parts in the silhouette and background clutter. The missing parts originate from similar colours in the appearance of the test subject and the background e.g., the test subject's dark pants against the dark wall of the lab. In contrast, the background clutter originates from other people in the background and the active infrared Qualisys cameras (the cameras are sensitive to near-infrared light and turn on after the background model was already generated).

The proposed shape from incomplete silhouette algorithm from Chapter 5 can also handle mistakes in the foreground/background segmentation. A missing part in the foreground has the same effect as an occluder between the camera and the test subject. Our method can generate a more complete reconstruction based on these incomplete silhouettes by using geometric reasoning. In Figure 7.3a we see the reconstruction of the standard SfS method. A substantial number of voxels are missing around the legs. In Figure 7.3b the reconstruction is shown from the proposed method from the same angle. We see that some voxel clusters (cells) are now also part of the reconstruction, such as part of the head (hair) and the lower legs. The different colours in this image represent different cells from the cell-based reconstruction (see Chapter 5).

The proposed reconstruction contains more voxels that are part of the person, and therefore the analysis may become more reliable. Figure 7.3c



(a) centroid tracking *(b) running movement*

Figure 7.4: *Spatio-temporal centroid tracking for gait analysis. The rectangle again indicates the frames where the test subject is inside the zone of maximum camera coverage.*

shows the difference in voxel count in the reconstructed shape between both methods during a run of the test subject. The red rectangle corresponds to the maximum camera coverage (red zone in Figure 7.1).

The reconstruction is used to derive several metrics such as the centre of mass, the step length, and the person's height. Each of these measurements can be calculated for each frame in the videos and therefore offer a detailed spatio-temporal analysis of the test object's movement. Figure 7.4a shows a model of walking/running movement. The person's centroid makes a sinusoidal movement, a trend we also see in the plot of the centroid of the reconstructed shape (unfiltered data points) in Figure 7.4b. From such an analysis, the number of steps can be extracted and also the average step length. A sinusoidal function can be fit with the data to cope with noise on the data points. The benefit of using the proposed shape from incomplete silhouettes reconstruction is limited because the centroid is the average position of all reconstructed voxels and since the reconstructed person is not missing an arm or some other body part that would cause asymmetry in the shape, the average centroid position still corresponds to the expected trend. Moreover, the shape-from-silhouttes has higher precision because it is very strict on which voxels to include in its 3D reconstruction. We also notice that the centroid from the proposed method is more noisy compared to the SfS's centroid.

Gait analysis is also an ideal use case for our 3D pose estimation method. From the data shown in the graphs of Figure 7.5, we derived a number of important parameters. Table 7.2 shows the numeric results of the important gait analysis parameters to describe the walking/running movement

	shape	pose est.	ground truth
Left cycle time	0.65 s	0.72 s	0.692 s
Left step length	0.73 m	0.76 m	0.729 m
Left step time		0.33 s	0.346 s
Left stance time		0.34 s	0.294 s
Left swing time		0.37 s	0.398 s
Right cycle time	0.65 s	0.70 s	0.692 s
Right step length	0.73 m	0.79 m	0.738 m
Right step time		0.37 s	0.346 s
Right stance time		0.31 s	0.300 s
Right swing time		0.38 s	0.392 s
Speed	2.12 m/s	2.11 m/s	2.136 m/s

Table 7.2: *Numerical result of the gait analysis. Results are shown for the shape reconstruction and the 3D human pose estimation. The analysis on the volumetric shape reconstruction can accurately determine some metrics. The results are less precise than the ground truth due to the limited voxel size of the reconstruction (1 cm) and frame rate (67 fps). The 3D human pose estimation can determine more metrics, but not all of them are as accurate as the floor plate because it is unclear when the foot is standing still.*

of the test subject. Note that the number of parameters that we derived using the shape reconstruction was limited and the cycle time are is derived from the centroid's position, not from the reconstruction of the feet because it is unclear when the feet touch the ground in the 3D reconstruction. We did, however, measure the step lengths from the 3D reconstruction (heel to heel).

The 3D human pose estimation has more detailed information and, therefore, we were able to estimate more parameters than using the volumetric shape reconstruction. However, it is unclear from the keypoint positions if a foot is decelerating, standing still, or accelerating. In the right graph of Figure 7.5, the Y coordinate (the direction in which the runner moves), is always slightly changing and the derivative function of the heel's y-coordinate are rarely 0. Therefore, the estimated swing and stance times are not reliable (a fixed threshold of 30 mm/s was used).

We compare both methods with the gold standard of the marker-based system that used the floor plate and Qualisys system to extract all parameters accurately. The results are similar for most parameters. Since the maximum frame rate of our system is only 67 fps, we cannot reach 0.001 s time precision but rather 0.015 s. Apart from the lower time accuracy, we see that both the shape reconstruction and 3D human pose estimation are rather accurate to perform such an analysis. Note

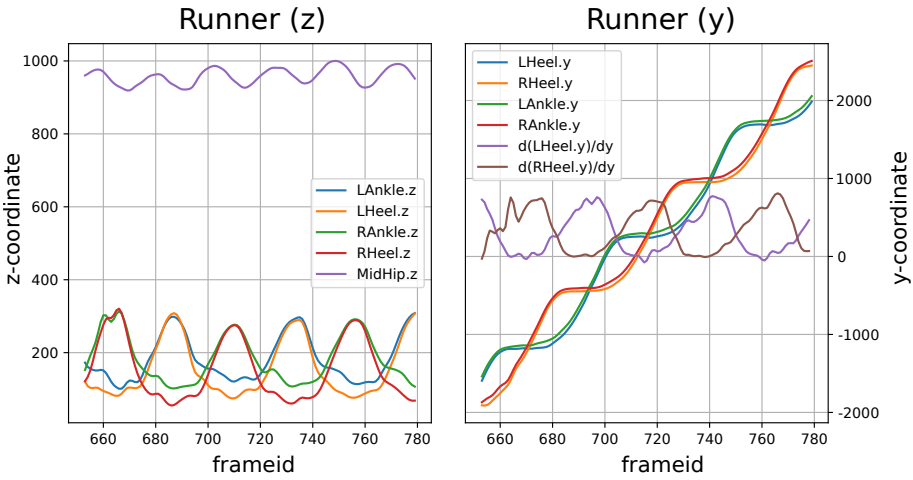


Figure 7.5: *Gait analysis based on 3D human pose estimation. We see repetitive patterns from the different steps of the runner. Note that the midhip is showing the expected sinusoidal shape.*

that we only need calibrated cameras and no instrumentation of markers on the test subjects to reach these results.

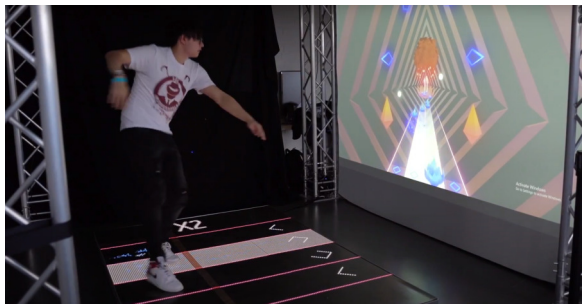
7.2 Game controller and rehabilitation (IPLAY)

Immersive games are becoming more popular, partly due to AR/VR systems such as the Oculus Rift, HTC Vive and other AR/VR headsets. Most commercial systems are limited to tracking the position of the player's head and hand positions. Therefore, the players' legs and feet are often not shown in games, or at least they will not correspond to the actual position of the player's legs and feet. This behaviour breaks the immersive experience.

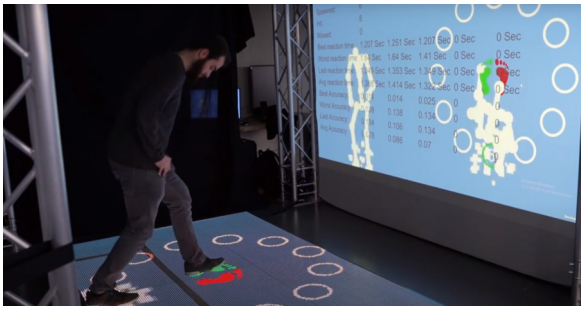
In the imec.icon project IPLAY, we researched the viability of a system consisting of LED floor tiles with pressure sensors and a wall projection for an immersive gaming experience. We used four cameras mounted around the field of play to track the players, creating an accurate markerless pose estimation system. The system ran in real-time at 20-25 fps on a powerful computer with two Nvidia 2080TI GPUs. The dedicated system streamed the reconstructed poses over a local area network to the game engine.

Two gaming applications were developed, which are controlled by the player's movement without any controllers (Figure 7.6). The first game

(a) Endless runner



(b) JAMZ



(c) Rehabilitation use case

Figure 7.6: *Two games and one rehabilitation application were developed by DAE in Kortrijk using our 3D human pose estimation in the IPLAY project. Teaser video: https://youtu.be/_6XeFguS7bI*

.

was an endless running application where a person needed to avoid virtual obstacles by dodging, jumping and crouching. The height of the lifted knees determined the avatar's speed in the game during the stationary running motion. A second game, JAMZ, exploited the features of the 3D tracking system even further by also using the person's hand positions during dancing movements. During the final event of the imec.ICON project, the IPLAY system was tried by several visitors and kept performing well. One of the biggest challenges during this demonstration was the numerous spectators visible in one or two camera views. However, the system still managed to determine which 2D poses from different views corresponded to the player. The system made no visible errors, which meant that the pose matching and keypoint positions were reliable for this application.

The imec.icon IPLAY project also focused on rehabilitation. The same system is used to assess if a patient is executing certain exercises correctly. Processing speed was less important for these applications, while

(a) Mariasteen *(b) CNHi*

Figure 7.7: *Examples of expert recordings at Mariasteen (first aid kit) and CNHi (central beam) for (semi)-automatic instruction extraction.*

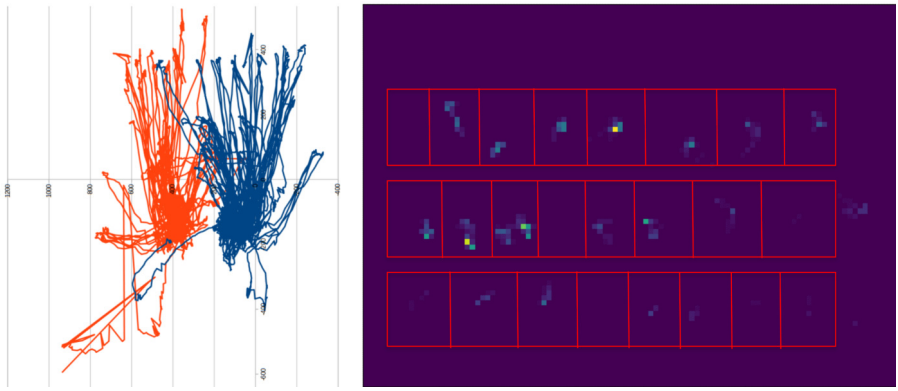
accuracy is more important than in gaming applications. We can achieve higher accuracy by feeding images with higher resolution to the pose estimator's neural network at the cost of higher processing time. However, the few centimetre accuracies that we reached with our system (see Section 6.6) was accurate enough to keep the same real-time performance for the rehabilitation application.

7.3 Expert recording (COSMO)

Our 3D pose estimation has also been used for behaviour analysis. Manufacturing companies use VR training to teach their employees how to perform specific tasks before performing these tasks for real. Alternatively, they support their employees with AR-based instructions during these particular tasks. In general, the more experienced the employee is, the less detailed these instructions need to be. In the COSMO project, these aspects are researched in detail.

Generating work instructions for AR and VR training is time-consuming. Our part in this project aims to (semi-)automatically generate these work instructions based on an expert recording. In such a recording, an expert shows how to perform a particular task. We capture the scene using different cameras and a microphone. We track the operator in the videos and reconstruct the expert's 3D pose to detect actions and events for the instruction generation.

Using speech recognition, the expert controls the recording and helps to segment different instructions and specify the tools and other objects involved in the task. In this project, we focus on two setups: one at Mariasteen and one at CNHi (Figure 7.7). The two use cases come with their challenges, which we will discuss in the following paragraph.



(a) Hand positions *(b) Picking rack*

Figure 7.8: *Hand positions of the operator and the automatic extraction of picking places using a heatmap of the hand locations in the first aid kit dataset at Mariasteen (bin = 2.5 cm).*

At Mariasteen, we focus on the assembly of a first aid kit. This use case is simple, in the sense that the genre of instruction is limited to roughly picking objects and placing them into the first aid kit, however we need to monitor the precise handling of the expert. Therefore, we installed six cameras around the expert to record his actions in detail. Detailed hand motions are of utmost importance, so we use OpenPose to detect the 21 keypoints per hand to assess the operator's action more precisely. By reconstructing these hand keypoints in 3D, using the same technique as the body poses, we can determine the location of specific picking spots automatically (Figure 7.8). We also assign which objects are located in each spot by using the speech recognition. Besides this, we also monitor changes in the first aid kit to determine where each object needs to be placed in the box and, hence, obtain the instruction set to assemble a first kit according to the expert recording.

At CNHi, we focus on the assembly of the central beam of a pick thresher. This assembly process is more complex compared to the first aid kit at Mariasteen. The operational area is about 5.5 m $\times$ 2.5 m. The expert executes more diverse actions such as sitting, drilling, walking, picking, and placing. To capture this process in detail, we used 11 GoPro cameras mounted around the reconstruction space. Four of these cameras are used to reconstruct the expert's pose, and the other cameras are used to detect events. We created a heatmap of the expert's hands to automatically find interesting areas where the expert executes most manipulations (Figure 7.9). By monitoring the area around the expert's hands in the different images, we detect tools and other objects. Like the

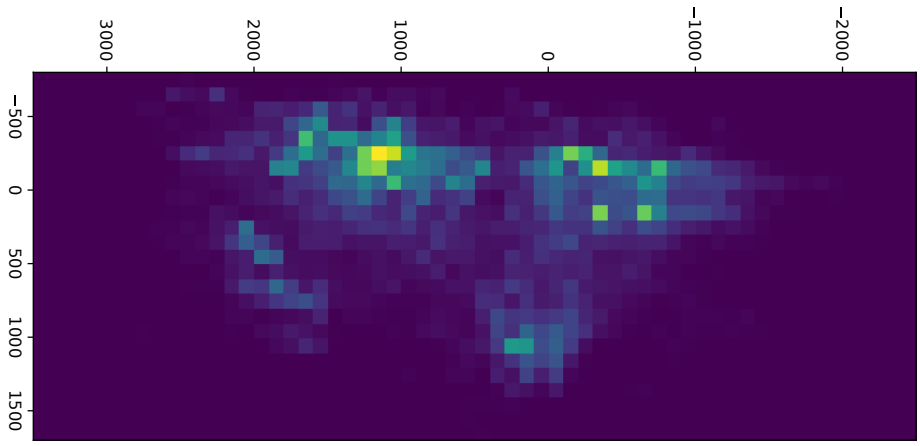


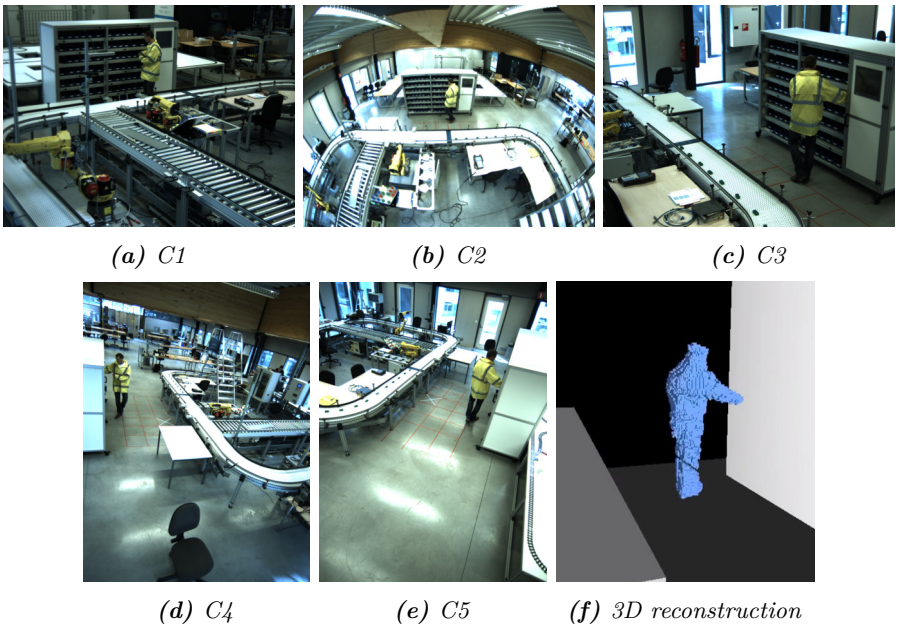
Figure 7.9: *Heatmap of the expert's hands in the scene (topview) to automatically determine the areas of interest (bin size = 10 cm). We choose the bin size four times larger than in the use case of the first aid kit of Mariasteen because the camera calibration is less accurate and the interesting regions in the scene are much larger.*

first aid kit, we also monitor the appearance changes on the central beam to detect where operations took place. Despite the background noise in the manufacturing hall of CNHi, we were able to record the expert's speech using a wireless microphone. The speech helps to determine which actions are performed with which object.

7.4 Assembly line analysis (Complexity)

Generating work instructions based on an expert recording is only one factory application where the proposed methods are helpful. When working with an assembly line, each work cycle needs to finish in time to avoid delays. Therefore, optimising these individual tasks is crucial. During the StarTT project Complexity in cooperation with Xiak (industrial automation centre of expertise Kortrijk), we built a test setup where five network cameras monitored the assembly area from different views (Figure 7.10). This assembly area consists of a conveyor belt and a storage rack. In each work cycle, the factory worker walks to the storage rack to fetch the parts and assembles them at the conveyor belt. Each camera recorded videos with a resolution of 1624 × 1234 px at 30 fps. A capture PC saved and synchronised the different video streams.

Machinery and other objects in such an environment often occlude the factory worker from a camera's perspective. In this dataset, occlusions



(a) C1 *(b) C2* *(c) C3*

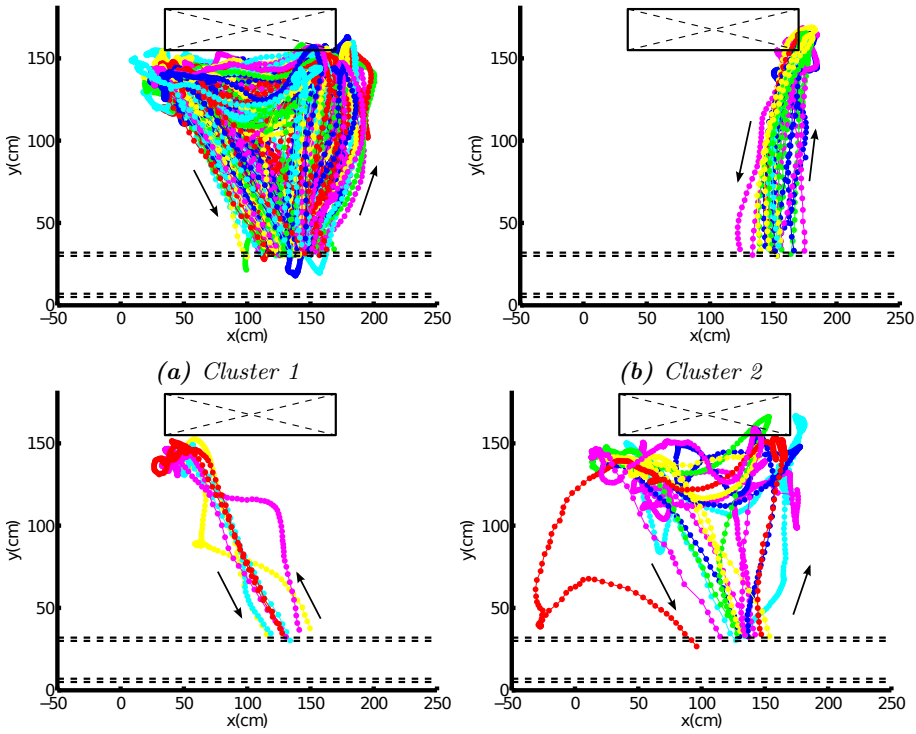
(d) C4 *(e) C5* *(f) 3D reconstruction*

Figure 7.10: *Complexity StarTT project: assembly line analysis. (a) to (e) show typical input images of the cameras. In (f) the 3D reconstruction is shown. Notice that the occluded lower limbs in (a) are also reconstructed.*

often happen in three out of five views (first row in Figure 7.10). However, our proposed shape reconstruction algorithm can handle these occlusions and create a volumetric shape reconstruction of the complete person.

The research conducted on this dataset analysed different work cycles executed based on the factory worker's trajectories. We cluster the work cycle trajectories to identify similar work cycles. For instance, the factory worker sometimes forgot a part and returned to the storage rack to fetch the correct parts. The number of data points in each trajectory is different, and therefore, we used dynamic time warping to compare trajectories. The temporal dissimilarity between trajectories was based on best velocity matching. The clustering method used a greedy clustering method using the dissimilarity measure to find patterns of the trajectory.

By manually checking the videos of this dataset, we find that the factory worker executed three different types of work cycles by visiting different areas of the storage rack: visiting the left and then the right corner of the storage rack; visiting just the left corner, and visiting just the right corner. The greedy clustering algorithm identified three main



(a) Cluster 1 *(b) Cluster 2*

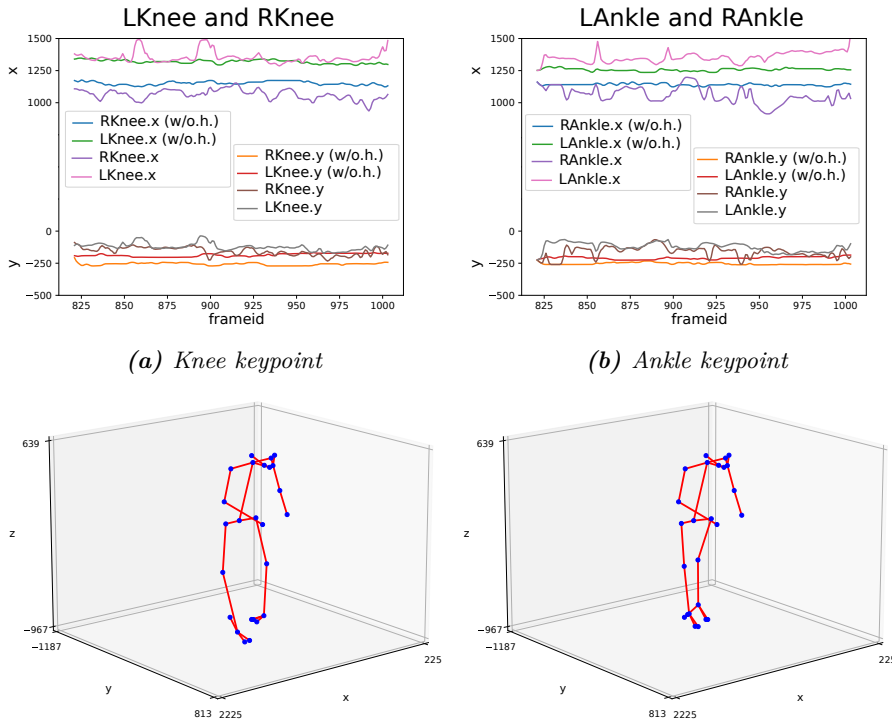
(c) Cluster 3 *(d) Spatial abnormalities*

Figure 7.11: *Clustered trajectories based on spatial dissimilarity. These results were published in the PhD of Xinzhe Xie [106].*

spatial clusters of trajectories. Occasionally, the workers pick up the tools and parts differently, for instance, visiting another area of the work station, visiting the storage rack from the right to the left corner, etc. These unusual executed work cycles lead to spatially abnormal trajectories. By clustering the trajectories, we can find the factory worker's moving patterns and the abnormalities of visiting the storage rack. This information is useful to optimise the work cycle.

A total of 124 trajectories were analysed, and different clusters were found (Figure 7.11). The trajectories are shown in different colours and assigned to three main work cycle clusters corresponding with 100, 13 and 4 trajectories separately, and 7 abnormalities. The main difference between these three types of executed work cycle is how the factory worker walked between the working station and the storage rack. The abnormalities are, in fact, separate clusters which are dissimilar to any other clusters in the dataset.

The results show that the trajectories are clustered into different pat-



(a) *Knee keypoint*

(b) *Ankle keypoint*

(c) *Without occlusion handling*

(d) *With occlusion handling*

Figure 7.12: *The shape reconstruction with occlusion handling is used to detect occluded keypoints and remove them from the 2D poses. In this test the operator is standing still. The stability of the triangulation in the occluded area in front of the conveyor belt improved significantly in (a) and (b) (w/o.h. = with occlusion handling). An example of the 3D reconstruction is shown in (c) without occlusion handling and (d) with occlusion handling. (d) shows a more realistic pose of the operator.*

terns successfully, and abnormalities are detected both spatially unfit to the prototypical routes and temporally different from other trajectories in the same spatial cluster. We conducted this research in cooperation with Xingzhe Xie and researchers from Xiak. The positional data extracted from the dataset was used in six publications [107, 108, 5, 4, 6].

This dataset provides an actual use case to test the reconstruction of the person's pose in the presence of occlusions. Occlusions are detected using our proposed 3D reconstruction method. The detected keypoints of the pose estimator in these occluded areas are removed. In this way, our 3D pose estimator can rely on visible points (not occluded) to generate a more reliable reconstruction. The results are shown in Figure 7.12 for the area in front of the conveyor belt where 3 out of 5 views are

occluded. This application shows how the combination of 3D human pose estimation and shape reconstruction with occlusion handling leads to a reliable monitoring system for human pose estimation in occluded environments.

7.5 Conclusion

This chapter gave an overview of some projects and applications of the proposed techniques in this book. We note that the proposed techniques functions as middleware for numerous applications. The game controller and rehabilitation use case is an example of our 3D pose estimation, added with an extra layer. In the expert recording use case, we noticed that the 3D pose estimation is helpful to understand the operator's actions in the scene and extract work instructions by the manipulations performed by the operator. The assembly analysis is a fine example of a use case where the occlusion detection based on shape reconstruction creates added value for the pose estimation by removing unreliable 2D keypoint detections in occluded regions.

We performed a basic analysis on the obtained shapes and pose reconstruction, but this analysis goes beyond the scope of this work. In essence, that process has already partly happened because data produced by this work has functioned as datasets for other researchers.

Chapter 8

Conclusions

Real-time 3D scene reconstruction becomes more and more important with the rise of autonomous cars and immersive games for AR/VR. Decades ago, 3D reconstruction from multiple camera views was only possible for static object because it took a lot of time to process the results. Nowadays, computers have become so powerful that the same can often be applied in real time. These developments have enabled the use of 3D reconstruction in a much broader field.

This manuscript focused on two aspects of 3D reconstruction: volumetric shape reconstruction and 3D human pose estimation. Shape reconstruction is useful for the volumetric analysis of objects/people e.g., collision avoidance between cars, the analysis of body shapes, etc. 3D human pose estimation accurately tracks a small number of 3D points, which enables pose analysis and as an extension, behaviour analysis. For a full scene understanding, both reconstruction types are combined.

To avoid that 3D reconstruction methods do not perform robustly in real-world circumstances, phenomena like occlusion need to be handled correctly as there is no guarantee that occlusion will not occur in most applications. Standard 3D reconstruction techniques are often limited in use because it is assumed that occlusion does not occur which breaks consistency assumptions between views. Depending on the application, it may not be much of an issue that one of the reconstructed objects is missing some parts, but this will nevertheless have an impact on the decisions that are based on the incomplete data. The methods presented in this manuscript are designed to handle incomplete data due to occlusion or erroneous segmented silhouettes and reconstruct a complete shape based on incomplete data.

While many of the building blocks (Chapters of this work) are well understood, there are still advancements to be made in each of them that affect the blocks that follow. Neural network-based segmentation algorithms are now outperforming traditional colour segmentation and foreground/background segmentation techniques, which in turn produce more reliable silhouettes for a silhouettes-based reconstruction algorithm, especially in challenging circumstances e.g., outdoor environments. These methods are computationally more expensive, but due to ever more powerful GPUs, some of these networks can already run in real time. Also the field of camera calibration is still evolving, even though the camera pinhole model is already decades old. The ever increasing image resolutions for instance enable more complicated calibration patterns to be used than with low resolution cameras.

The structure of this manuscript was organised to reflect the entire workflow from camera images to 3D reconstruction. However, since both volumetric shape reconstruction and 3D human pose estimation are discussed, there are two stories to be told. To understand volumetric shape reconstruction or 3D pose estimation the reader needs to be familiar with perspective transformation from 3D world to image to reverse this process in the 3D reconstruction (see Chapter 2). The volumetric shape reconstruction is based on silhouette extraction as discussed in Chapter 3, while the volumetric shape reconstruction itself is explained in Chapter 4. The advanced occlusion handling is explained in Chapter 5. 3D pose estimation is laid out in Chapter 6. Chapter 7 is of interest to both reconstruction methods as it goes deeper into a number of applications in which both methods are useful and also showcases a combination where occlusion detection based on volumetric shape reconstruction helps to reconstruct human poses more reliable.

In the body of this manuscript, we have discussed a number of different improvements to pre-existing literature and listed a number of contributions to the scientific world. The main contributions of this work are:

- a practical tool to calibrate multiple cameras using checkerboard patterns and calibration objects, and to analyse a camera network based on camera coverage;
- an analysis of different silhouette extraction techniques such as colour segmentation, foreground/background segmentation and neural network segmentation;
- a real-time voxel-based implementation of shape-from-silhouettes;

- a reliable cross-view pose correspondence method based on maximal cliques combined with a method to handle common pose estimator issues such as mislabelled limb keypoints;
- a complete end-to-end system to capture and process multiple video feeds to reconstruct shapes from incomplete silhouettes;
- a cell-based reconstruction method that can handle incomplete silhouettes originating from occlusion or silhouette segmentation errors;
- the combination of volumetric shape reconstruction and 3D human pose estimation to improve scene understanding;
- the valorization of the methods in several applications developed during different projects in cooperation with a number of companies and academic partners.

In total, the research during this PhD resulted in 7 peer-reviewed journal papers, of which 1 as first author. Moreover 17 peer-reviewed conference papers were presented at international conferences, of which 7 as first author and 10 as co-author (see Appendix A).

Appendices

Appendix A

List of Publications

A.1 A1 publications

1. M. Slembrouck, P. Veelaert, D. Van Cauwelaert, D. Van Hamme, and W. Philips. Cell-based shape reconstruction from incomplete silhouettes. *Integrated Computer-Aided Engineering*, 26(3):257–271, 2019. ISSN 1069-2509

2. K. Bauters, J. Cottyn, D. Claeys, M. Slembrouck, P. Veelaert, and H. van Landeghem. Automated work cycle classification and performance measurement for manual work stations. *Robotics and computer-aided manufacturing*, 51, 2018. ISSN 0736-5845

3. J. Guan, F. Deboeverie, M. Slembrouck, D. Van Haerenborgh, D. Van Cauwelaert, P. Veelaert, and W. Philips. Extrinsic calibration of camera networks based on pedestrians. *Sensors*, 16(5), 2016. ISSN 1424-8220

4. J. Li, M. Slembrouck, F. Deboeverie, A. M. Bernardos, J. A. Besada, P. Veelaert, H. Aghajan, J. R. Casar, and W. Philips. Handheld pose tracking using vision-inertial sensors with occlusion handling. *Journal of Electronic Imaging*, 25(4):14, 2016. ISSN 1017-9909

5. J. Niño Castañeda, A. Frias Velazquez, B. B. Nyan, M. Slembrouck, J. Guan, G. Debard, B. Vanrumste, T. Tuytelaars, and W. Philips. Scalable semi-automatic annotation for multi-camera person tracking. *IEEE Transactions on image processing*, 25(5):2259–2274, 2016. ISSN 1057-7149

6. J. Guan, F. Deboeverie, M. Slembrouck, D. Van Haerenborgh, D. Van Cauwelaert, P. Veelaert, and W. Philips. Extrinsic calibration of camera networks using a sphere. *Sensors*, 15(8):18985–19005, 2015. ISSN 1424-8220

7. B. B. Nyan, F. Deboeverie, M. Eldib, J. Guan, X. Xie, J. Niño Castañeda, D. Van Haerenborgh, M. Slembrouck, S. Van de Velde, H. Steendam, P. Veelaert, R. Kleihorst, H. Aghajan, and W. Philips. Human mobility monitoring in very low resolution visual sensor network. *Sensors*, 14(11):20800–20824, 2014. ISSN 1424-8220

A.2 Conference publications

1. M. Slembrouck, H. Luong, J. Gerlo, K. Schütte, D. Van Cauwelaert, D. De Clercq, B. Vanwanseele, P. Veelaert, and W. Philips. Multiview 3d markerless human pose estimation from OpenPose skeletons. In *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS'20)*, volume 12002, pages 166–178. Springer, 2020. ISBN 9783030406042

2. B. B. Nyan, M. Slembrouck, P. Veelaert, and W. Philips. Distributed multi-class road user tracking in multi-camera network for smart traffic applications. In *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS'20)*, volume 12002, pages 517–528. Springer, 2020. ISBN 9783030406042

3. G. Allebosch, M. Slembrouck, S. Roegiers, H. Luong, P. Veelaert, and W. Philips. Foreground background segmentation in front of changing footage on a video screen. In *Proceedings of Advanced concepts for intelligent vision systems (ACIVS'18)*, volume 11182, pages 175–187. Springer International Publishing, 2018. ISBN 9783030014483

4. M. Slembrouck, P. Veelaert, D. Van Hamme, D. Van Cauwelaert, and W. Philips. Cell-based approach for 3d reconstruction from incomplete silhouettes. In *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS'17)*. Springer, 2017

5. M. Slembrouck, D. Van Cauwelaert, P. Veelaert, and W. Philips. Shapes-from-silhouettes based 3d reconstruction for athlete evaluation during exercising. In *Abstracts of Science and Engineering Conference on Sports Innovations (SECSI'16)*, page 2, 2016

6. J. Li, M. Slembrouck, F. Deboeverie, A. M. Bernardos, J. A. Besada, P. Veelaert, H. Aghajan, W. Philips, and J. R Casar. A hybrid pose tracking approach for handheld augmented reality. In *Proceedings of the 9th International Conference on Distributed Smart Cameras (ICDSC'15)*, pages 7–12. ACM, 2015. ISBN 978-1-4503-3681-9

7. J. Li, B. Goossens, M. Slembrouck, F. Deboeverie, P. Veelaert, H. Aghajan, W. Philips, and J. R Casar. Demo: a new 360-degree immersive game controller. In *Proceedings of the 9th International Conference on Distributed Smart Cameras (ICDSC'15)*, pages 201–202. ACM, 2015. ISBN 978-1-4503-3681-9

8. X. Xie, D. Van Cauwelaert, M. Slembrouck, K. Bauters, J. Cottyn, D. Van Haerenborgh, H. Aghajan, P. Veelaert, and W. Philips. Abnormal work cycle detection based on dissimilarity measurement of trajectories. In *Proceedings of the 9th International Conference on Distributed Smart Cameras (ICDSC'15)*, page 6. ACM, 2015. ISBN 978-1-4503-3681-9

9. M. Slembrouck, J. Niño Castañeda, G. Allebosch, D. Van Cauwelaert, P. Veelaert, and W. Philips. High performance multi-camera tracking using shapes-from-silhouettes and occlusion removal. In *Proceedings of the 9th International Conference on Distributed Smart Cameras (ICDSC)*, pages 44–49. ACM, 2015. ISBN 978-1-4503-3681-9

10. M. Slembrouck, D. Van Cauwelaert, P. Veelaert, and W. Philips. Shape-from-silhouettes algorithm with built-in occlusion detection and removal. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP'15)*. SCITEPRESS, 2015

11. X. Xie, J. De Vylder, D. Van Cauwelaert, P. Veelaert, W. Philips, H. Aghajan, M. Slembrouck, D. Van Haerenborgh, H. Van Landeghem, K. Bauters, J. Cottyn, and H. Vervaeke. Average track estimation of moving objects using ransac and dtw. In *Proceedings of the 8th International Conference on Distributed Smart Cameras (ICDSC'14)*, page 6. ACM, 2014. ISBN 978-1-4503-2925-5

12. K. Bauters, H. Van Landeghem, M. Slembrouck, D. Van Cauwelaert, and D. Van Haerenborgh. An automated work cycle classification and disturbance detection tool for assembly line work

stations. In *Proceedings of the International Conference on Informatics in Control, Automation and Robotics (ICINCO'14)*, volume 2, page 7, 2014. ISBN 9789897580406

13. M. Slembrouck, D. Van Cauwelaert, D. Van Hamme, D. Van Haerenborgh, P. Van Hese, P. Veelaert, and W. Philips. Self-learning voxel-based multi-camera occlusion maps for 3d reconstruction. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP'14)*, page 8. SCITEPRESS, 2014

14. P. Veelaert, M. Slembrouck, and D. Van Haerenborgh. Concurrency relations of digital planes. In *Proceedings of the 17th IAPR International Conference Discrete Geometry for Computer Imagery (DGCI'13)*, volume 7749, pages 347–359. Springer Berlin Heidelberg, 2013. ISBN 9783642370663

15. K. Bauters, H. Van Landeghem, D. Van Haerenborgh, M. Slembrouck, D. Van Cauwelaert, P. Veelaert, and W. Philips. Multi-camera complexity assessment system for assembly line work stations. In *Proceedings of the European Concurrent Engineering Conference (ECEC'13)*, page 5, 2013

16. D. Van Hamme, M. Slembrouck, D. Van Haerenborgh, D. Van Cauwelaert, P. Veelaert, and W. Philips. Parameter-unaware auto-calibration for occupancy mapping. In *Proceedings of the 7th international Conference on Distributed Smart Cameras (ICDSC'13)*, pages 49–54. IEEE, 2013. ISBN 9781479921669

17. M. Slembrouck, M. Heyvaert, D. Van Cauwelaert, D. Van Hamme, P. Veelaert, and W. Philips. Time complexity of traditional vision algorithms on a block-based image processor (BLIP). In *Proceedings of the Sixth international conference on distributed smart cameras (ICDSC'2012)*, page 6. IEEE, 2012. ISBN 9781450317726

Appendix B

Events

Parts of this dissertation and some side projects were presented at several events. In most cases, we built a live demonstration to engage the visitors.

1. Boetiek Techniek 2012
   Description: Science fair for young children to learn about technology. David Van Hamme, Dimitri Van Cauwelaert and I created a virtual ball game in which a thrown ball was tracked and caught by a net. However, we used the virtual continuation of the trajectory to knock over the virtual cones.
   Date: 12 May 2012
   Location: Ghent, Belgium

2. Art and D 2012
   Description: Cooperation between researchers and artists, a program by IBBT (now imec). In cooperation with Vadim Vosters, a demo was created in which a person's shadow was recorded and showed in another place to create the sense of a remote presence. It worked bidirectionally and created natural interactions.
   Date: 7 November 2012
   Location: ICC, Ghent, Belgium

3. Boetiek Techniek 2013
   Description: Science fair for young children to learn about technology. David Van Hamme, Dimitri Van Cauwelaert and I created a virtual pong game where the bats could be controlled by hand on a table.
   Date: 18 May 2013
   Location: Ghent, Belgium

4. iBoot 2015
   Description: Valorisation program of iMinds (now imec) to research the possibility to create a business around the technology presented in this thesis together with Dimitri Van Cauwelaert.
   Date: 26 April - 1 May 2015
   Location: London, United Kingdom

5. iMinds the Conference 2016
   Description: International conference of iMinds for companies and research institutes. We developed a live demo to illustrate occlusion in a multi-camera setup and use it to create 3D models of the occluders in the scene.
   Date: 28 April 2016
   Location: Square Brussels Meeting Center, Brussels, Belgium

6. Sports Innovation Congress 2017
   Description: Congress to watch and listen to the future of sports. The IPLAY project was demonstrated at this event, and we showed our markerless motion analysis in a separate boot.
   Date: 17 October 2017
   Location: Julien Saelens Sport Vlaanderen domain, Bruges, Belgium

7. Future Summits 2019
   Description: International conference of imec for companies and research institutes. A live demo was developed to illustrate the occlusion of the 3D joint and shape reconstruction in real-time.
   Date: 14-15 May 2019
   Location: Antwerp, Belgium

Bibliography

[1] G. Allebosch, D. Van Hamme, F. Deboeverie, P. Veelaert, and W. Philips. C-EFIC: Color and edge based foreground background segmentation with interior classification. In *Communications in Computer and Information Science*, volume 598, pages 433–454. Springer, 2016. ISBN 978-3-319-29971-6.

[2] G. Allebosch, M. Slembrouck, S. Roegiers, H. Luong, P. Veelaert, and W. Philips. Foreground background segmentation in front of changing footage on a video screen. In *Proceedings of Advanced concepts for intelligent vision systems (ACIVS'18)*, volume 11182, pages 175–187. Springer International Publishing, 2018. ISBN 9783030014483.

[3] O. Barnich and M. Van Droogenbroeck. ViBe: A universal background subtraction algorithm for video sequences. *IEEE Transactions on Image Processing*, 20(6):1709–1724, June 2011. doi:10.1109/TIP.2010.2101613.

[4] K. Bauters, H. Van Landeghem, D. Van Haerenborgh, M. Slembrouck, D. Van Cauwelaert, P. Veelaert, and W. Philips. Multi-camera complexity assessment system for assembly line work stations. In *Proceedings of the European Concurrent Engineering Conference (ECEC'13)*, page 5, 2013.

[5] K. Bauters, H. Van Landeghem, M. Slembrouck, D. Van Cauwelaert, and D. Van Haerenborgh. An automated work cycle classification and disturbance detection tool for assembly line work stations. In *Proceedings of the International Conference on Informatics in Control, Automation and Robotics (ICINCO'14)*, volume 2, page 7, 2014. ISBN 9789897580406.

[6] K. Bauters, J. Cottyn, D. Claeys, M. Slembrouck, P. Veelaert, and H. van Landeghem. Automated work cycle classification and

performance measurement for manual work stations. *Robotics and computer-aided manufacturing*, 51, 2018. ISSN 0736-5845.

[7] B. E. Bayer. Color imaging array, July 1976. US Patent 3,971,065.

[8] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann. Blazepose: On-device real-time body pose tracking. *arXiv preprint arXiv:2006.10204*, 2020.

[9] P. A. Beardsley, A. Zisserman, and D. W. Murray. Sequential updating of projective and affine structure from motion. *International journal of computer vision*, 23(3):235–259, 1997.

[10] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee. Yolact: real-time instance segmentation. In *Proceedings of the International Conference on Computer Vision (ICCV'19)*, pages 9157–9166. IEEE, 2019.

[11] J.-Y. Bouguet et al. *Visual methods for three-dimensional modeling*. Citeseer, 1999.

[12] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

[13] D. H. Brown. Decentering distortion of lenses. In *Photogrammetric Engineering*, volume 32, pages 444—462, 1966.

[14] D. A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, and D. Kim. Shake'n'sense: reducing interference for overlapping structured light depth cameras. In *Proceedings of the Conference on Human Factors in Computing Systems (SIGCHI'12)*, pages 1933–1936, 2012.

[15] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'17)*, 2017.

[16] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*, 2018.

[17] H. Chen, K. Sun, Z. Tian, C. Shen, Y. Huang, and Y. Yan. Blendmask: Top-down meets bottom-up for instance segmentation. *arXiv preprint arXiv:2001.00309*, 2020.

[18] 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 *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)*. IEEE, 2003.

[19] K. Cheung, S. Baker, J. Hodgins, and T. Kanade. Markerless human motion transfer. In *Proceedings of the 2nd International Symposium on 3D Data Processing Visualization and Transmission (3DPVT'04)*, pages 373–378, Sep. 2004.

[20] W. Coste. *Basketball Plays, Tricks and Gimmicks*. Xlibris US, 2010. ISBN 9781453553411. URL https://books.google.be/books?id=8qgvZ_ANWBYC.

[21] D. F. Dementhon and L. S. Davis. Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, 15(1-2): 123–141, 1995.

[22] L. Díaz-Más, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and R. Medina-Carnicer. Shape from silhouette using dempster–shafer theory. *Pattern Recognition*, 43(6):2119–2131, 2010.

[23] L. Díaz-Más, F. J. Madrid-Cuevas, R. Muñoz-Salinas, A. Carmona-Poyato, and R. Medina-Carnicer. An octree-based method for shape from inconsistent silhouettes. *Pattern Recognition*, 45(9):3245–3255, 2012.

[24] P. Ding and Y. Song. Robust object tracking using color and depth images with a depth based occlusion handling and recovery. In *Proceedings of the 12th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD'15)*, pages 930–935. IEEE, 2015.

[25] S. Donné. *From light rays to 3D models*. PhD thesis, Ghent University, 2018.

[26] O. Drbohlav, editor. *Next view planning for shape from silhouette*, 02 2003.

[27] Y. Du, Y. Wong, Y. Liu, F. Han, Y. Gui, Z. Wang, M. Kankanhalli, and W. Geng. Marker-less 3d human motion capture with monocular image sequence and height-maps. In *Proceedings of the European Conference on Computer Vision (ECCV'16)*, pages 20–36. Springer, 2016.

[28] G. Duan, H. Ai, and S. Lao. A structural filter approach to human detection. In *Proceedings of the European Conference on Computer Vision (ECCV'10)*, pages 238–251. Springer, 2010.

[29] A. Elhayek, O. Kovalenko, P. Murthy, J. Malik, and D. Stricker. Fully automatic multi-person human motion capture for vr applications. In *Proceedings of the 15th International Conference EuroVR (EuroVR'18)*, 2018.

[30] P. Favaro, A. Duci, Y. Ma, and S. Soatto. On exploiting occlusions in multiple-view geometry. In *Proceedings of the 9th International Conference on Computer Vision (ICCV'03)*, pages 479–486. IEEE, 2003.

[31] R. B. Girshick, P. F. Felzenszwalb, and D. A. Mcallester. Object detection with grammar models. In *Advances in Neural Information Processing Systems*, pages 442–450, 2011.

[32] N. Goyette, P. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. Changedetection.net: A new change detection benchmark dataset. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition Workshops (CVPRW'12)*, pages 1–8. IEEE, 2012.

[33] S. Grünwedel, N. Petrovic, L. Jovanov, J. Niño Castañeda, A. Pizurica, and W. Philips. Efficient foreground detection for real-time surveillance applications. *Electronics Letters*, 49(18): 1143–1144, 2013. ISSN 0013-5194.

[34] J. Guan. *Calibration of camera networks : kalibratie van cameranetwerken*. PhD thesis, Ghent University, 2017.

[35] J. Guan, F. Deboeverie, M. Slembrouck, D. Van Haerenborgh, D. Van Cauwelaert, P. Veelaert, and W. Philips. Extrinsic calibration of camera networks using a sphere. *Sensors*, 15(8): 18985–19005, 2015. ISSN 1424-8220.

[36] J. Guan, F. Deboeverie, M. Slembrouck, D. Van Haerenborgh, D. Van Cauwelaert, P. Veelaert, and W. Philips. Extrinsic calibration of camera networks based on pedestrians. *Sensors*, 16(5), 2016. ISSN 1424-8220.

[37] L. Guan, S. Sinha, J.-S. Franco, and M. Pollefeys. Visual hull construction in the presence of partial occlusion. In *Proceedings*

of the 3rd International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06), pages 413–420. IEEE Computer Society, 2006. doi:10.1109/3DPVT.2006.138.

[38] G. Haro and M. Pardàs. Shape from incomplete silhouettes based on the reprojection error. *Image and Vision Computing*, 28(9): 1354–1368, 2010.

[39] R. Hartley and F. Schaffalitzky. ℓ_∞ minimization in geometric reconstruction problems. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'04)*, volume 1, pages I–I. IEEE, 2004.

[40] R. Hartley and P. Sturm. Triangulation. In V. Hlaváč and R. Šára, editors, *Computer Analysis of Images and Patterns*, pages 190–197, Berlin, Heidelberg, 1995. Springer. ISBN 978-3-540-44781-8.

[41] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the International Conference on Computer Vision (ICCV'17)*, pages 2961–2969. IEEE, 2017.

[42] J. Heikkilä and O. Silvén. A four-step camera calibration procedure with implicit image correction. In *Proceedings of Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97)*, pages 1106–1112. IEEE, 1997.

[43] M. Hofmann and D. M. Gavrila. Multi-view 3d human pose estimation combining single-frame recovery, temporal integration and model adaptation. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'09)*, pages 2214–2221, June 2009.

[44] Hudl. Hudl technique - analyze athlete performance. https://www.hudl.com/products/technique, 2021. Accessed 2021-01-27.

[45] F. Huo, E. Hendriks, P. Paclik, and A. H. J. Oomes. Markerless human motion capture and pose recognition. In *Proceedings of the 10th Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS'09)*, pages 13–16, May 2009.

[46] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, and B. Schieke. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. In *European Conference on Computer Vision (ECCV'16)*, 2016.

[47] A. Kadkhodamohammadi and N. Padoy. A generalizable approach for multi-view 3d human pose regression. *Machine Vision and Applications*, 32(1):1–14, 2021.

[48] S. B. Kang, R. Szeliski, and J. Chai. Handling occlusions in dense multi-view stereo. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'01)*, volume 1, pages I–103. IEEE, 2001.

[49] H. Kim, R. Sakamoto, I. Kitahara, T. Toriyama, and K. Kogure. Robust foreground segmentation from color video sequences using background subtraction with multiple thresholds. In *Proceedings of the Korea-Japan Workshop on Pattern Recognition*, pages 188–193, 2006.

[50] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International journal of Computer Vision*, 38(3):199–218, 2000.

[51] J.-L. Landabaso, M. Pardàs, and J. R. Casas. Shape from inconsistent silhouette. *Computer Vision and Image Understanding*, 112 (2):210–224, 2008.

[52] A. Laurentini. The visual hull concept for silhouette-based image understanding. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):150–162, Feb 1994. ISSN 0162-8828. doi:10.1109/34.273735.

[53] K. M. Lee and C.-C. J. Kuo. Shape reconstruction from photometric stereo. In *Proceedings of Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'92)*, pages 479–484, 1992. doi:10.1109/CVPR.1992.223147.

[54] B. Li, W. Hu, T. Wu, and S.-C. Zhu. Modeling occlusion by discriminative and-or structures. In *Proceedings of the International Conference on Computer Vision (ICCV'13)*, pages 2560–2567. IEEE, 2013.

[55] J. Li, B. Goossens, M. Slembrouck, F. Deboeverie, P. Veelaert, H. Aghajan, W. Philips, and J. R Casar. Demo: a new 360-degree immersive game controller. In *Proceedings of the 9th International Conference on Distributed Smart Cameras (ICDSC'15)*, pages 201–202. ACM, 2015. ISBN 978-1-4503-3681-9.

[56] J. Li, M. Slembrouck, F. Deboeverie, A. M. Bernardos, J. A. Besada, P. Veelaert, H. Aghajan, W. Philips, and J. R Casar. A hybrid pose tracking approach for handheld augmented reality. In *Proceedings of the 9th International Conference on Distributed Smart Cameras (ICDSC'15)*, pages 7–12. ACM, 2015. ISBN 978-1-4503-3681-9.

[57] J. Li, M. Slembrouck, F. Deboeverie, A. M. Bernardos, J. A. Besada, P. Veelaert, H. Aghajan, J. R. Casar, and W. Philips. Handheld pose tracking using vision-inertial sensors with occlusion handling. *Journal of Electronic Imaging*, 25(4):14, 2016. ISSN 1017-9909.

[58] J. Liu, S. Sridharan, C. Fookes, and T. Wark. Optimal camera planning under versatile user constraints in multi-camera image processing systems. *IEEE Transactions on Image Processing*, 23 (1):171–184, 2013.

[59] W. E. Lorensen and H. E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *ACM siggraph computer graphics*, 21(4):163–169, 1987.

[60] S. Mallick. Pose detection comparison : wrnchai vs OpenPose. https : / / www . learnopencv . com / pose-detection-comparison-wrnchai-vs-openpose, 2019. Accessed 2020-12-09.

[61] M. Mathias, R. Benenson, R. Timofte, and L. Van Gool. Handling occlusions with franken-classifiers. In *Proceedings of the International Conference on Computer Vision (ICCV'13)*, pages 1505–1512. IEEE, 2013.

[62] W. Matusik, C. Buehler, and L. McMillan. Polyhedral visual hulls for real-time rendering. In *Proceedings of the Eurographics Workshop on Rendering Techniques (EGSR'01)*, pages 115–125. Springer, 2001.

[63] N. F. Maunder and G. de Jager. *Virtual View Synthesis using Visual Hulls*. PhD thesis, University of Cape Town, 2005.

[64] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, H.-P. Seidel, W. Xu, D. Casas, and C. Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):44, 2017. doi:10.1145/3072959.3073596.

[65] P. Merriaux, Y. Dupuis, R. Boutteau, P. Vasseur, and X. Savatier. A study of vicon system positioning performance. *Sensors*, 17: 1591, 07 2017. doi:10.3390/s17071591.

[66] P. S. Milne. *Visual hulls for volume estimation: a fast marching cubes based approach*. PhD thesis, University of Cape Town, 2005.

[67] G. Nagymate and R. Kiss. Application of optitrack motion capture systems in human movement analysis a systematic literature review. *Recent Innovations in Mechatronics*, 5, 07 2018. doi:10.17667/riim.2018.1/13.

[68] J. Niño Castañeda, A. Frias Velazquez, B. B. Nyan, M. Slembrouck, J. Guan, G. Debard, B. Vanrumste, T. Tuytelaars, and W. Philips. Scalable semi-automatic annotation for multi-camera person tracking. *IEEE Transactions on image processing*, 25(5): 2259–2274, 2016. ISSN 1057-7149.

[69] B. B. Nyan, F. Deboeverie, M. Eldib, J. Guan, X. Xie, J. Niño Castañeda, D. Van Haerenborgh, M. Slembrouck, S. Van de Velde, H. Steendam, P. Veelaert, R. Kleihorst, H. Aghajan, and W. Philips. Human mobility monitoring in very low resolution visual sensor network. *Sensors*, 14(11):20800–20824, 2014. ISSN 1424-8220.

[70] B. B. Nyan, M. Slembrouck, P. Veelaert, and W. Philips. Distributed multi-class road user tracking in multi-camera network for smart traffic applications. In *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS'20)*, volume 12002, pages 517–528. Springer, 2020. ISBN 9783030406042.

[71] OpenVINO. Openvino™ toolkit overview. https://docs.openvinotoolkit.org/latest/index.html, 2021. Accessed 2021-09-10.

[72] W. Ouyang and X. Wang. A discriminative deep model for pedestrian detection with occlusion handling. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR'12)*, pages 3258–3265. IEEE, 2012.

[73] W. Ouyang, X. Zeng, and X. Wang. Partial occlusion handling in pedestrian detection with a deep model. *IEEE Transactions on Circuits and Systems for Video Technology*, 26(11):2123–2137, Nov 2016. doi:10.1109/TCSVT.2015.2501940.

[74] W. Phothong, T.-C. Wu, J.-Y. Lai, D. W. Wang, C.-Y. Liao, and J.-Y. Lee. Fast and accurate triangular model generation for the shape-from-silhouette technique. *Computer-Aided Design and Applications*, 14(4):436–449, 2017.

[75] L. Pishchulin, E. Insafutdinov, S. Tang, B. Andres, M. Andriluka, P. Gehler, and B. Schiele. Deepcut: Joint subset partition and labeling for multi person pose estimation. In *Conference on Computer Vision and Pattern Recognition (CVPR'16)*. IEEE, 2016.

[76] E. Prados and O. Faugeras. Shape from shading. In *Handbook of mathematical models in computer vision*, pages 375–388. Springer, 2006.

[77] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings in Advances in Neural Information Processing Systems*, pages 91–99, 2015.

[78] B. Rosenhahn, U. G. Kersting, J. K. Smith, A. W. qnd Gurney, T. Brox, and R. Klette. A system for marker-less human motion estimation. In *Proceedings of the joint Pattern Recognition Symposium*, pages 230–237, Berlin, Heidelberg, 2005. Springer. ISBN 978-3-540-31942-9.

[79] J. Saboune and F. Charpillet. Markerless human motion capture for gait analysis. *arXiv*, 2005.

[80] A. Saharan. Creating a human pose estimation application with NVIDIA DeepStream. https://developer.nvidia.com, 2020. Accessed 2021-09-10.

[81] D. Senior. *Qualisys Track Manager: User Manual*. National Research Council Canada. Institute for Ocean Technology, 2004. doi:10.4224/8896115.

[82] 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 *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'11)*, pages 1297–1304. IEEE, 2011.

[83] M. Slembrouck, M. Heyvaert, D. Van Cauwelaert, D. Van Hamme, P. Veelaert, and W. Philips. Time complexity of traditional vision

algorithms on a block-based image processor (BLIP). In *Proceedings of the Sixth international conference on distributed smart cameras (ICDSC'2012)*, page 6. IEEE, 2012. ISBN 9781450317726.

[84] M. Slembrouck, D. Van Cauwelaert, D. Van Hamme, D. Van Haerenborgh, P. Van Hese, P. Veelaert, and W. Philips. Self-learning voxel-based multi-camera occlusion maps for 3d reconstruction. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP'14)*, page 8. SCITEPRESS, 2014.

[85] M. Slembrouck, J. Niño Castañeda, G. Allebosch, D. Van Cauwelaert, P. Veelaert, and W. Philips. High performance multi-camera tracking using shapes-from-silhouettes and occlusion removal. In *Proceedings of the 9th International Conference on Distributed Smart Cameras (ICDSC)*, pages 44–49. ACM, 2015. ISBN 978-1-4503-3681-9.

[86] M. Slembrouck, D. Van Cauwelaert, P. Veelaert, and W. Philips. Shape-from-silhouettes algorithm with built-in occlusion detection and removal. In *Proceedings of the International Conference on Computer Vision Theory and Applications (VISAPP'15)*. SCITEPRESS, 2015.

[87] M. Slembrouck, D. Van Cauwelaert, P. Veelaert, and W. Philips. Shapes-from-silhouettes based 3d reconstruction for athlete evaluation during exercising. In *Abstracts of Science and Engineering Conference on Sports Innovations (SECSI'16)*, page 2, 2016.

[88] M. Slembrouck, P. Veelaert, D. Van Hamme, D. Van Cauwelaert, and W. Philips. Cell-based approach for 3d reconstruction from incomplete silhouettes. In *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS'17)*. Springer, 2017.

[89] M. Slembrouck, P. Veelaert, D. Van Cauwelaert, D. Van Hamme, and W. Philips. Cell-based shape reconstruction from incomplete silhouettes. *Integrated Computer-Aided Engineering*, 26(3):257–271, 2019. ISSN 1069-2509.

[90] M. Slembrouck, H. Luong, J. Gerlo, K. Schütte, D. Van Cauwelaert, D. De Clercq, B. Vanwanseele, P. Veelaert, and W. Philips. Multiview 3d markerless human pose estimation from OpenPose skeletons. In *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS'20)*, volume 12002, pages 166–178. Springer, 2020. ISBN 9783030406042.

[91] D. Snow, P. Viola, and R. Zabih. Exact voxel occupancy with graph cuts. In *Proceedings of Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'00)*, volume 1, pages 345–352. IEEE, 2000.

[92] P. L. St-Charles, G. A. Bilodeau, and R. Bergevin. Subsense: A universal change detection method with local adaptive sensitivity. *IEEE Transactions on Image Processing*, 24(1):359–373, Jan 2015. ISSN 1057-7149. doi:10.1109/TIP.2014.2378053.

[93] J. Starck and A. Hilton. Surface capture for performance-based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007.

[94] J. Sun, Y. Li, S. B. Kang, and H.-Y. Shum. Symmetric stereo matching for occlusion handling. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 399–406. IEEE, 2005.

[95] T. Svoboda, D. Martinec, and T. Pajdla. A convenient multi-camera self-calibration for virtual environments, 2005.

[96] TechSmith. Coach's eye - sports video analysis app. https://www.coachseye.com, 2021. Accessed 2021-01-27.

[97] S. Ullman. The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153):405–426, 1979.

[98] D. Van Hamme. *Robust ego-localization using monocular visual odometry.* PhD thesis, Ghent University, 2016.

[99] D. Van Hamme, M. Slembrouck, D. Van Haerenborgh, D. Van Cauwelaert, P. Veelaert, and W. Philips. Parameter-unaware auto-calibration for occupancy mapping. In *Proceedings of the 7th international Conference on Distributed Smart Cameras (ICDSC'13)*, pages 49–54. IEEE, 2013. ISBN 9781479921669.

[100] P. Veelaert, M. Slembrouck, and D. Van Haerenborgh. Concurrency relations of digital planes. In *Proceedings of the 17th IAPR International Conference Discrete Geometry for Computer Imagery (DGCI'13)*, volume 7749, pages 347–359. Springer Berlin Heidelberg, 2013. ISBN 9783642370663.

[101] K. Wegner, O. Stankiewicz, and M. Domański. Occlusion handling in depth estimation from multiview video. In *Proceedings of the International Conference on Signals and Electronic Systems (ICSES'14)*, pages 1–4. IEEE, 2014.

[102] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh. Convolutional pose machines. In *Proceedings of the Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'16)*, 2016.

[103] S. Wu, G. Zhang, M. Zhu, T. Jiang, and F. Neri. Geometry based three-dimensional image processing method for electronic cluster eye. *Integrated Computer-Aided Engineering*, 25:1–16, 01 2018. doi:10.3233/ICA-180564.

[104] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. Accessed 2020-12-09.

[105] Y. Xiang and S. Savarese. Object detection by 3d aspectlets and occlusion reasoning. In *Proceedings of the International Conference on Computer Vision Workshops (ICCVW'13)*, pages 530–537. IEEE, 2013. doi:10.1109/ICCVW.2013.75.

[106] X. Xie. *Human and vehicle trajectory analysis.* PhD thesis, Ghent University, 2016.

[107] X. Xie, J. De Vylder, D. Van Cauwelaert, P. Veelaert, W. Philips, H. Aghajan, M. Slembrouck, D. Van Haerenborgh, H. Van Landeghem, K. Bauters, J. Cottyn, and H. Vervaeke. Average track estimation of moving objects using ransac and dtw. In *Proceedings of the 8th International Conference on Distributed Smart Cameras (ICDSC'14)*, page 6. ACM, 2014. ISBN 978-1-4503-2925-5.

[108] X. Xie, D. Van Cauwelaert, M. Slembrouck, K. Bauters, J. Cottyn, D. Van Haerenborgh, H. Aghajan, P. Veelaert, and W. Philips. Abnormal work cycle detection based on dissimilarity measurement of trajectories. In *Proceedings of the 9th International Conference on Distributed Smart Cameras (ICDSC'15)*, page 6. ACM, 2015. ISBN 978-1-4503-3681-9.

[109] Xilinx. Xilinx edge AI solution. https://www.xilinx.com/applications/industrial/analytics-machine-learning.html, 2019. Accessed 2021-09-10.

[110] Y. Xiu, J. Li, H. Wang, Y. Fang, and C. Lu. Pose Flow: Efficient online pose tracking. In *Proceedings of the British Machine Vision Conference (BMVC'18)*, 2018.

[111] Y. Yemez and F. Schmitt. 3d reconstruction of real objects with high resolution shape and texture. *Image and Vision computing*, 22(13):1137–1153, 2004.

[112] Z. Zhang. Flexible camera calibration by viewing a plane from unknown orientations. In *Proceedings of the 7th International Conference on Computer Vision*, volume 1, pages 666–673. IEEE, 1999. doi:10.1109/ICCV.1999.791289.

[113] C. L. Zitnick and T. Kanade. A cooperative algorithm for stereo matching and occlusion detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(7):675–684, 2000.

[114] Z. Zivkovic. Improved adaptive gaussian mixture model for background subtraction. In *Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04)*, volume 2, pages 28–31 Vol.2, Aug 2004. doi:10.1109/ICPR.2004.1333992.

Use case where multiple cameras capture Short Track Speed Skaters and reconstruct their pose for technical analysis, allowing the coach to focus on the training itself.