May 2014

Use of Microsoft Kinect in a dual camera setup for action recognition applications

Omar Ghassan Kayal
The University of Western Ontario

Supervisor
Jagath Samarabandu
The University of Western Ontario

Graduate Program in Electrical and Computer Engineering

A thesis submitted in partial fulfillment of the requirements for the degree in Master of Engineering Science

© Omar Ghassan Kayal 2014

Follow this and additional works at: https://ir.lib.uwo.ca/etd
Part of the Other Computer Engineering Commons

Recommended Citation
https://ir.lib.uwo.ca/etd/2072

This Dissertation/Thesis is brought to you for free and open access by Scholarship@Western. It has been accepted for inclusion in Electronic Thesis and Dissertation Repository by an authorized administrator of Scholarship@Western. For more information, please contact tadam@uwo.ca, wlsadmin@uwo.ca.
USE OF MICROSOFT KINECT IN A DUAL CAMERA SETUP FOR ACTION RECOGNITION APPLICATIONS

(Thesis format: Monograph)

by

Omar Kayal

Graduate Program in Electrical and Computer Engineering

A thesis submitted in partial fulfillment of the requirements for the degree of Masters of Engineering Science

The School of Graduate and Postdoctoral Studies
The University of Western Ontario
London, Ontario, Canada

© Omar Kayal 2014
Abstract

Conventional human action recognition methods use a single light camera to extract all the necessary information needed to perform the recognition. However, the use of a single light camera poses limitations which can not be addressed without a hardware change. In this thesis, we propose a novel approach to the multi camera setup. Our approach utilizes the skeletal pose estimation capabilities of the Microsoft Kinect camera, and uses this estimated pose on the image of the non-depth camera. The approach aims at improving performance of image analysis of multiple camera, which would not be as easy in a typical multiple camera setup. The depth information sharing between the camera is in the form of pose projection, which depends on location awareness between them, where the locations can be found using chessboard pattern calibration techniques. Due to the limitations of pattern calibration, we propose a novel calibration refinement approach to increase the detection distance, and simplify the long calibration process. The two tests performed demonstrate that the pose projection process performs with good accuracy with a successful calibration and good Kinect pose estimation, however not so with a failed one. Three tests were performed to determine the calibration performance. Distance calculations were prone to error with a mean accuracy of 96% under 60cm difference, and dropping drastically beyond that, and a stable orientation calculation with mean accuracy of 97%. Last test also proves that our new refinement approach improves the outcome of the projection significantly with a failed pattern calibration, and allows for almost double the camera difference detection of about 120cm. While the orientation mean calculation accuracy achieved similar results to pattern calibration, the distance was less so at around 92%, however, it did maintain a stable standard deviation, while the pattern calibration increased as distance increased.

Keywords: Kinect, depth sensor, multi camera, dual camera, calibration, pose projection
Acknowledgements

I would like to start by thanking my supervisor Dr. Jagath Samarabandu for giving me the privilege to work with him on his research starting in 2012. I am grateful for his valuable mentor-ship, advice, motivation, and financial support during my program. I am appreciative of the vast knowledge he shared with me, his time that he so generously offered, and his dedication to help me reach my goals with the completion of my Masters degree. With his guidance, I have expanded my research capabilities, gained priceless knowledge, and improved professionally. I have enjoyed my time, time which will be missed. Thank you Dr. Samarabandu.

I would also like to thank my parents, Ghassan and Rouba, and siblings, Rayan and Baraa for their great support, and for believing in me throughout the tough years. I could not have done it without you.

I would also like to thank Dr. Abdelkader Ouda for his guidance and great mentor-ship during my Teaching Assistance work. You have helped me greatly, and overlooked my shortcomings, which I sincerely appreciate.

I would like to thank my colleague Ali for his help on countless occasions relating my research work, course work, and Teaching Assistance work. You have helped me greatly, and I will not forget that. I would also like to thank my colleague Akila for directing me on my research work, and his willingness to always help. You are a great photographer by the way. I would also like to thank my colleague Dwayne for involving me with his research study, it was a great experience. Lastly, I would like to thank my colleague Raymond, for his support, and for his amazing demonstration of meteorite rocks and Mars talk. Good luck with your research. You guys are awesome, and I thank you all.

Lastly, I would like to thank my great friends Ahmad Turk, Amin El-Nagger, Bandar Aldhafeeri, Feras Obeid, Hassan Ouda, Nader Hallak, Saqib Dadabhoy and Thabet Marwa for their support, and being there for me throughout my Masters experience. It would have been a lot harder without you guys. Thank you.
To my loving parents Ghassan and Rouba
## Contents

Certificate of Examination ........................................ ii

Abstract ................................................................. ii

List of Figures .......................................................... x

List of Tables .......................................................... xiii

List of Appendices ...................................................... xiii

1 Introduction .............................................................. 1

1.1 Background ......................................................... 1

1.2 Motivation ............................................................ 3

1.3 Problem statement ................................................... 3

1.4 Summary of contribution .......................................... 4

1.5 Outline ............................................................... 5

2 Literature review ...................................................... 6

2.1 Brief introduction to the Action Recognition model ............ 7

2.2 Low Level: Core technology ..................................... 10

2.2.1 Object segmentation ........................................... 10

Static camera ............................................................. 11

Moving camera ........................................................... 13

2.2.2 Feature extraction ............................................... 14
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.2.3 Activity detection and classification</td>
<td>15</td>
</tr>
<tr>
<td>2.3 Mid level: Human Activity Recognition Systems</td>
<td>17</td>
</tr>
<tr>
<td>2.4 High level: Applications</td>
<td>18</td>
</tr>
<tr>
<td>2.5 Motivation for the proposed approach</td>
<td>18</td>
</tr>
<tr>
<td>2.5.1 Single camera feature extraction and action recognition</td>
<td>18</td>
</tr>
<tr>
<td>2.5.2 Stereo camera setup</td>
<td>20</td>
</tr>
<tr>
<td>2.5.3 Multi view camera setup</td>
<td>20</td>
</tr>
<tr>
<td>2.6 Microsoft Kinect</td>
<td>21</td>
</tr>
<tr>
<td>2.7 Multi view Kinect setup</td>
<td>22</td>
</tr>
<tr>
<td>2.8 Calibration</td>
<td>23</td>
</tr>
<tr>
<td>2.8.1 Brief overview</td>
<td>23</td>
</tr>
<tr>
<td>2.8.2 Single camera Calibration</td>
<td>25</td>
</tr>
<tr>
<td>3 Method Architecture and Justification</td>
<td>27</td>
</tr>
<tr>
<td>3.1 Setup overview</td>
<td>28</td>
</tr>
<tr>
<td>3.1.1 Brief Introduction</td>
<td>28</td>
</tr>
<tr>
<td>3.1.2 Design</td>
<td>29</td>
</tr>
<tr>
<td>3.2 Stereo-calibration</td>
<td>34</td>
</tr>
<tr>
<td>3.3 Pose estimation</td>
<td>36</td>
</tr>
<tr>
<td>3.3.1 Depth extraction</td>
<td>36</td>
</tr>
<tr>
<td>3.3.2 Skeletal extraction</td>
<td>37</td>
</tr>
<tr>
<td>3.3.3 Skeletal Projection</td>
<td>38</td>
</tr>
<tr>
<td>3.4 Calibration refinement</td>
<td>40</td>
</tr>
<tr>
<td>3.4.1 Overview</td>
<td>40</td>
</tr>
<tr>
<td>3.4.2 Angle correction</td>
<td>41</td>
</tr>
<tr>
<td>3.4.3 Distance correction</td>
<td>42</td>
</tr>
<tr>
<td>3.4.4 Center Point extraction</td>
<td>44</td>
</tr>
<tr>
<td>3.5 Software</td>
<td>47</td>
</tr>
</tbody>
</table>
4 Testing and results

4.1 Stereo Calibration ................................................................. 54
  4.1.1 Test 1: Pattern recognition ........................................... 55
    Test 1 results and discussion ............................................. 56
  4.1.2 Test 2: Measurement accuracy vs pattern count .................... 58
    Results: ........................................................................ 58
  4.1.3 Test 3: Measurement Accuracy vs distance and angle between the cameras ................................................................. 60
    Maximum Distance (Test 3a) ............................................... 61
    Maximum Angle (Test 3b) .................................................. 61
    Results: ........................................................................ 61
  4.1.4 Discussion of test results .................................................. 64

4.2 Pose estimation and projection ................................................ 66
  4.2.1 Test 4: Kinect Pose estimation ......................................... 66
    Results: ........................................................................ 67
  4.2.2 Test 5: Pose projection .................................................... 69
    Results: ........................................................................ 69

4.3 Calibration refinement .............................................................. 71
  4.3.1 Test 6: Calibration refinement ......................................... 72
    Maximum Distance (Test 6a) ............................................... 72
    Maximum Angle (Test 6b) .................................................. 72
    Results: ........................................................................ 73

5 Conclusion and future work .......................................................... 77

  5.1 Summary ........................................................................ 77

  5.2 Future work ................................................................... 78
    5.2.1 Improved tracking ....................................................... 78
    5.2.2 Stereo Self calibration .................................................. 79
List of Figures

1.1 Examples of different use of recognition. a) Blob tracking, a form that detects the entire body and represents it as a box, useful in tracking players in a playing field b) Body controller, a very recent addition to the game industry, uses the tracking of body parts as the controller to enhance the gaming experience c) Medical applications, useful in physiotherapy and health monitoring d) Another example of blob tracking, Surveillance ................................. 2

2.1 Object segmentation [15] ................................................................. 9
2.2 Features extraction categories [15] .................................................. 9
2.3 Action recognition algorithms [15] .................................................... 10
2.4 Action recognition System [15] ........................................................ 11
2.5 Background subtraction ................................................................. 13
2.6 Pose estimation Examples [15] ........................................................ 15
2.7 Action learning and classification [15] .............................................. 17
2.8 Distance Space [4] ................................................................. 19
2.9 Pin hole camera .............................................................................. 23
2.10 Pin hole camera Model (Excluding the skew) ................................. 24
2.11 ........................................................................................................ 26

3.1 3 dimensional Euler angles ............................................................. 29
3.2 Kinect axis directions and Euler angle orientation. The Y axis is towards the top of the camera, the X axis is to the left, and the Z axis is in the Kinect camera lens direction ................................................................. 30
3.3 A Dual Camera setup between the Kinect and a standard camera. The light triangle signifies the Kinect’s view area, and the darker signifies the standard camera’s view area. Both triangles intersect in the view overlap section.

3.4 Skeleton model [20]

3.5 Process breakdown

3.6 Stereo Calibration step, the two pictures were taken at the same time

3.7 Shotton et al [25] depth image and ground truth body parts

3.8 Kinect and camera in world axis. The figure shows the equivalent depth value of Kinect with respect to the other camera’s local axis. The Kinect is capable of depth values up to 3.5m, however, the other camera is not restricted with this value.

3.9 Finding the angle in the calibration correction step

3.10 Finding the distance difference between the two cameras

3.11 3D to 2D projection

3.12 Silhouettes and edges

3.13 Silhouettes and edges

3.14 Process outline

3.15 Main class

3.16 kinectCam and camera class

3.17 skeletalProjection class

3.18 calibCamera and stereoCalib classes

3.19 recalibration class

3.20 CreateButtonLayout class

3.21 UML class diagram

4.1 Distance consistency performance

4.2 Angle consistency performance

4.3 Distance performance
5.1 Ayazoglu’s el al.’s [1] method in action. As can be seen from the image, the trajectory of the car tracked by one camera is mapped by the other camera with the occluded view ........................................ 79

5.2 a) Articulated human body model b) Position of the 5 coplanar points during Mid stance [16] ................................................ 80

5.3 a) First camera b) Second camera [5] ................................. 81

A.1 Kinect camera specs [19] .............................................. 87

A.2 Infrared Dot pattern .................................................. 88

B.1 Orthographic projection view 1 ...................................... 90

B.2 Orthographic projection result 2 .................................... 90

B.3 Orthographic projection view 3 ...................................... 92

B.4 Orthographic projection view 4 ...................................... 92
List of Appendices

Appendix A Kinect camera ......................................................... 87
Appendix B 3D to 2D Projection .................................................. 89
Chapter 1

Introduction

1.1 Background

With technology integration into our lives becoming more and more defined and established than ever before, it is becoming crucial to find better ways to communicate with technology. This exchange comes in different forms ranging from well defined mouse and keyboard inputs, to more complex audio, video, and image input. However, advancement in computing has made it possible to process large volumes of data in a short amount of time, giving rise to better machine recognition of the less defined data.

Computer vision has been a hot topic for many years, due its huge potential and application base, and with the advancement in computing power, its now more possible than ever before. Computer vision is used in surveillance, gaming, sport tracking, health care monitoring and much more as shown in figure 1.1. However, designing a system that will work in a specific application is a challenging task, and has to be carefully designed to be capable of handling unexpected changes.

Human action recognition in computer vision is becoming increasingly in high demand, and the need for a reliable, marker-less systems has never been so critical. However, developing a method that can cope with a broad range of actions, especially in complex scenarios, is still a
challenge. Recent advances in computer vision and pattern recognition had made it possible to recognize more complex action, making for more reliable systems.

Figure 1.1: Examples of different use of recognition. a) Blob tracking, a form that detects the entire body and represents it as a box, useful in tracking players in a playing field b) Body controller, a very recent addition to the game industry, uses the tracking of body parts as the controller to enhance the gaming experience c) Medical applications, useful in physiotherapy and health monitoring d) Another example of blob tracking, Surveillance

The demand for recognition system started a boom of new approaches, each with its own set of strengths and flaws, trying to address different scenarios and different situation, where each design would work well in the environment it is designed for. There exist the single camera paradigm, designed for simpler recognitions, where each approach differs based on the features used. There are stereo camera, using the difference in disparity between two images to find the depth data, adding a more capable feature list than normal RGB cameras. Depth camera are

\(^4\text{http://www.cs.ubc.ca/~shervmt/}\)
\(^4\text{http://www.goodhousekeeping.com/product-reviews/research-institute/kinect}\)
\(^4\text{http://www.physioatthelodge.co.uk/physiotherapy/}\)
\(^4\text{http://privacysos.org/node/411/}\)
also gaining usage traction, with cameras like the Time of Flight camera ToF, and the recent introduction of the widely available Kinect camera. Finally, the multiple camera paradigm, were multiple camera are used in unison to expand the available feature list for more capable systems.

1.2 Motivation

The goal of the thesis is to build the foundation for a more capable system that can work in any computer vision application, including human action recognition, tracking, and health care both for physiotherapy and an individual’s well-being monitoring, and improve the likelihood of a successful system by providing many visual resources at the disposal of the user at a low cost.

We achieve this through the use of dual camera system, with one of the cameras having a depth sensing capability. By partial view sharing, the depth data can be shared between both cameras, plus the ability to extract 3D from the said depth data in both cameras.

1.3 Problem statement

The use of non-stereo multi cameras in computer vision is well established. However, they do suffer from performance issues and low result accuracy when computing depth data [6], and are not always the right choice for human body pose estimation. In addition, they typically exhibit no form of information sharing. As such, there is a need to improve performance of image analysis algorithms under the multiple camera banner. Additionally, there needs to be a way to share information between both cameras to help better in the recognition of objects. This help could come from the utilization of occlusion handling, expanded view, and view variance from different camera views.

To make use of information sharing, both cameras have to be location aware, and that awareness can be achieved using calibration methods. However, these methods are error prone,
and can be difficult to use for different people, especially product consumers. Finding a better and easier way of determining the location is hence a necessity.

1.4 Summary of contribution

In this thesis, we introduce a novel approach of integrating Kinect, a depth camera, with a second non-depth camera into a dual camera setup. By doing so, we are eliminating the need to compute the disparity between image differences to find the depth data, where the process is both inaccurate and resource heavy. Our approach and contribution is the addition of pose projection to the depth dual camera setup, a method where we take the depth based estimated human pose from the Kinect camera, and applying it onto the proper human body segmentation of the non-depth camera’s image plane. Such a system requires spatial knowledge of the cameras, which is achieved using a stereo calibration method explained in the following chapters.

The use of the Microsoft Kinect camera grants a cheaper alternative to more expensive, time of flight cameras, with comparable results. The approach will be useful in human action recognition applications, and this work augments ongoing research in this area within the research group. The approach is aimed at providing improved visual resources of both depth and color data for action recognition applications. In addition, the approach gains the advantages of a dual camera system, and a depth capable system.

Our second contribution will be the introduction of a calibration refinement step aimed at finding the relative geometric, distance and angle, difference parameters between the dual cameras. The proposed method builds on top of the stereo based chessboard pattern calibration method, where the calibration refinement aims to correct any calibration result errors. The aim of the proposed method is to reduce the time needed to find the geometric parameters of a dual camera setup that would otherwise have been time consuming should it have been performed using pattern calibration.
Lastly, we will conduct six tests to test out the proposed approach. The first three tests will provide insight onto the calibration method by Zhang et al. [32]. This includes distance and angle detection limitations, accuracy of the methods calculations, and the reliability based on the number of patterns used. The fourth test will cover the pose estimation of the depth data by Shotton et al. [25]. The fifth test will test the performance of the proposed projection of the estimated pose. Lastly, test six will test the limitations of orientation and distance calculations, and its accuracy of the proposed calibration refinement process, which will be compared in performance to the pattern calibration method.

1.5 Outline

The remainder of the thesis is as follows. In chapter 2, related work will be explored. A quick overview of action recognition will be first presented, then we will explore related work, present the limitation of each method, and build up to the reason for our proposed method. The full approach description is provided in chapter 3. It will cover the setup, depth data pose estimation, chessboard pattern calibration, and finally, calibration refinement. In chapter 4, we will test and verify the method in chapter 3. This also includes testing the limitations of the approach. Lastly, chapter 5 will be the main conclusion to the thesis. This includes closing statements, and possible future work.
Chapter 2

Literature review

The method proposed in this thesis is part of a bigger project. So to better understand the motivation behind the research done, we will have to provide an insight at the future goal. A person does not appreciate being watched, even for medical reasons. To help serve the senior community better, we have proposed a project that aims at developing a fully automated first response medical monitoring system. The system would involve a full body pose recognition system that would recognize irregular actions, such as falling down, lying still for long periods of time, measuring irregular heart beats through an infrared bionic scanner (The second installment of the Kinect camera is capable of doing that), etc. Since the system is meant for home use, a multi camera setup would be ideal since it would cover more view, and provide better occlusion handling capabilities. Such a system would fall under the action recognition model. As a system built for recognition, it would make sense to explore the action recognition field, even through it is not directly realized in the main contribution of this thesis.

The ability of a camera to give real time 3D information allows for greater system interaction possibilities than a 2D one would. In addition to making the person segmentation and identification easier, an extra dimension would improve the recognition capabilities by having more data available for action classification. With a dual camera system, including the Kinect, the system would add the benefits of both a depth capable system and a multi camera system.
2.1 Brief introduction to the Action Recognition model

We take this setup a step further by giving the non-depth camera depth like capabilities. The availability of depth give rise to more accurate pose estimation opportunities, image segmentation and processing. In a home setup, we are more interested in the pose of the person than raw depth data for recognition, the reason for which will be explain in more detail under the action recognition model section 2.1. So by applying our contribution, we aim at providing pose for both views of the cameras.

The original pose estimation method built for the Kinect was developed for game use, assuming proper facing of the camera at all times, which means the estimated pose is not as accurate otherwise. In a home monitoring system, this would not be the case. Granted, the Kinect pose estimation method’s ability to generate a proper pose for postures not facing the camera is accurate to a certain degree, however it will fail to estimate the location of an occluded limb, even by the body’s own self occlusion. Finding a better pose estimation is a solution, however its not in the scope of this thesis. Our aim is to achieve pose projection between both cameras.

2.1 Brief introduction to the Action Recognition model

Visual based human recognition in computer vision is divided into many sub-topics such as action recognition, facial expression recognition and movement behavior recognition. Action recognition is defined as the process of naming different actions, usually using simple actions verbs, through the use of sensory observations. An action is classified as a human based sequence of movements that signify the performance of a task. An action can be thought of as a four-dimensional object, covering both spatial and temporal information. It is not necessarily limited to four-dimensions, where audio can also be used to classify said action. However, we will only be focusing on the computer vision aspect of action recognition, and as such, audio and other types of data input will be ignored.

Generally, there are three different levels of representation to human action recognition
Chapter 2. Literature review

[15]:

- Low level: Core Algorithms
- Mid level: Human Activity Recognition systems
- High level: Applications

The first low level part of the structure encompass the main technology build to handle action labeling. The process is responsible for object segmentation, feature extraction, and action detection and classification. In human action recognition, the human object is the target of object segmentation stage, the characteristics of which depend on the application requirement. Such characteristics include body silhouette, body motion, colors, poses, or anything unique enough that can be used. The object segmentation is implemented on every frame in a video stream, be that a prerecorded or live camera feed, to extract the required object. The segmentation process is dependent on the camera mode being used, either a static or a moving camera.

In the static camera mode, the camera is fixed in a specific location at a fixed angle, while the mobile camera mode is used in robot specific applications, or anything that requires a moving camera. Each mode employs different segmentation type due to the changing background model. As a result, the static camera mode is easier to handle, since the background is known, and background subtraction can be relatively easier to perform, however by no means trivial, due to high sensitively to slight illumination changes, and complex models such as mixture of Gaussians are needed to get reasonable results. The mobile camera mode uses different types of segmentation, such as temporal different, or optical flow, both of which are more complicated to implement.

The extracted segmentation is then represented by a set of features. Features can be though of a dimensionally reduced form of the segmentation data where only the needed relevant information is used. Features in general can be categorized into four groups, space-time volume,
2.1. Brief introduction to the Action Recognition model

Figure 2.1: Object segmentation [15]

frequency transform, local descriptors and body modeling as shown in figure 2.2. In our approach, we utilize the body modeling feature extraction category since the Kinect’s depth data segments provide the opportunity to get relatively accurate 3D model features.

Figure 2.2: Features extraction categories [15]

In the last stage of the low level core technology, the extracted features are analyzed by an action detection algorithm, and classified into action labels equating to the various recognized human actions. The type of algorithms used vary for different types of features and/or different
real world application needs. The list of algorithms are shown in figure 2.3.

![Figure 2.3: Action recognition algorithms](image)

In section 2.3, a brief overview of the mid level human activity recognition systems will be presented to show how each of the general configurations work. The configure covers single person activity recognition, multiple people interaction and crowd behaviors, and abnormal activity recognition. Finally, in the third and last stage of the process, the activity recognition is applied depending on the application field. Figure 2.4 summarizes the human action recognition process.

### 2.2 Low Level: Core technology

The core technology is the foundation of the action recognition system. Its split into three different sections, object segmentation, feature extraction and finally action recognition. Under the health monitoring design, the core technology would cover the pose estimation of the body, and the recognition of the different actions performed.

#### 2.2.1 Object segmentation

Object segmentation, as the name suggests, is the automatic process of separating the relevant foreground object from the background. In addition to action recognition, this process is used
2.2. Low Level: Core Technology

Figure 2.4: Action recognition System [15]

in tracking, image processing, and many other computer vision applications. In action recognition, the target object can be the full human body or just part of it such as a limb. The type of object segmentation used depends on the mobility of the camera, either a stationary camera with a fixed angle such as the ones used in healthcare, a stationary camera with a varying angle, such as pan-tilt cameras used in surveillance, and completely mobile camera, such as the ones used in robotics vehicle mounts. The more mobile a camera is, the harder the segmentation. In a stationary camera, the background is static and does not change. If the background is predetermined, any changes to the pixel values would indicate the foreground, making it easier to segment than in the case of a mobile camera where the background changes, and the camera trajectory has to be found. As for mobile cameras, no predetermined background model can be used leading to more complicated object segmentation.

**Static camera**

The static camera mode is easier to implement than the mobile camera mode. The reason for which is that the background is not static and does not change. This mode is most suitable
for health care applications and gaming. Generally, object segmentation on a static camera is done using background subtraction. Background subtraction is used for its simplicity and efficiency [23]. Ideally, background subtraction can be done by getting the absolute value of the subtracted image pixel values from the background model and the main image stream with the needed foreground. Any value greater than zero would be the background such as shown in figure 2.5a and 2.5b respectively. However, this is not usually the case since the background in the main image stream is never exactly the same the recorded background model, which results in background patches classified as foreground. This background inconsistency could be due to small illumination changes, camera image sensor quantization rounding, or just noise. A simple solution would be to use a threshold value instead of the zero mark to determine if the pixel location is a background pixel or a foreground pixel. Any result value below the threshold is a background pixel, and anything above is the foreground. However, this sometimes presents more problem than it solves.

Consider a person wearing clothing close in color to the background. If the difference in color between the background and the clothing is less than the assigned threshold, then a foreground pixel could be classified as a background one which would result in rough and fractional foreground objects such as the one shown in figure 2.5c.

To solve this problem, many authors have came up with different solutions, each one with its set of strength and weaknesses, built to satisfy a certain application base. For example, Cucchiara et al. [7] discriminates objects, ghosts and shadows from a video using a statistical assumption with object level knowledge of the moving objects, its apparent ghost and shadow. A different approach, Wren et al. [28] models the color distribution of each pixel in the background with a Gaussian with a full covariance matrix, which maps different foreground textures depending on the mean and covariance of a point. More complicated approaches, such as ones that use the Gaussian Mixture model and the Expectation-Minimization algorithm are also available for more complicated scenarios, but that goes beyond the scope of this thesis. For the sake of this thesis, a simple understanding of background subtraction is sufficient.
Moving camera

Moving cameras are cameras with a dynamic location and angle. Even cameras with a fixed location and variable angle, such as pan-tilt cameras used in surveillance, as considered moving cameras. Such cameras are used in robotics, vehicle mounted cameras, and countless others. Unlike a stationary camera, the background in a moving camera is always changing, so segmentation used in stationary cameras would never work in moving cameras. This makes the segmentation process of mobile cameras far more complicated since both the motion of the
camera and the motion of the foreground object have to be taken into account. This requires some form of motion decomposition to separate the motion of the camera and the motion of the object. Two well used techniques are temporal difference and optical flow. Simply, both techniques try to measure the motion of the scene between consecutive frames, isolate the background depending on the analyzed motion and find the required foreground. In the scene of a moving camera, the closer an object is to the camera, the faster the displacement of that object is compared to the background.

Our approach will utilize the static camera mode, which means we will be using static camera segmentation methods. As a design built to be used for home monitoring, it would be pointless to complicate things and use a mobile camera. In addition to that, a mobile segmentation framework used in a multi camera setup would require far more performance resource usage since constant location update is required.

### 2.2.2 Feature extraction

In the second stage of the low core technology level of the human action recognition model, the feature extraction is responsible for extracting important characteristics of the image frames and represented in a systematic way as features to work with the right recognition algorithm. The type feature extraction used directly effects the performance of recognition, so it’s always critical to chose the right type. Feature can be global or local. Global features such as space-time volumes and discrete Fourier transform are extracted by considering the whole image. However, global features are sensitive to occlusions and viewpoint variation [15]. Local features such as local descriptors and Scale Invariant Feature Transform (SIFT) features, solve this problem by considering local patches in the image as local features. This makes them more robust to noise, as well as scale and rotation. Other methods include modeling the human body which require pose estimation and body tracking techniques. Pose is the combined position and orientation of an object with respect to a camera. These body models can be further converted to lower dimensions to improve performance and refine recognition. See section 2.5
for more details.

Part of feature extraction is the human pose estimation. Human pose estimation techniques enable the development of accurate 2D or 3D representations of the human body. Such as the ones shown in figure 2.6.

### 2.2.3 Activity detection and classification

The final stage of the low core technology level of the action recognition model is action detection and classifications. After a frame in a video is segmented, the segmentation is converted into a feature that can be used by the selected action classification algorithm. It’s an ongoing research, and a huge topic on what kind of feature works best with what kind of recognition. The selection of the features used depends on the applications that the action recognition processing is going to be used on, and the suitable classification algorithm based on research results is chosen. For example, let’s compare the difference between two application scenarios with stationary cameras. In the first scenario, a fall detection system, the feature can be the center data of the segmented human silhouette. If the person was to fall, the center data would shift from a higher position to a lower position. A K-Nearest neighbor classification algorithm would be enough to determine what class of action, either standing up or falling down, the action
belongs to. In the other scenario, the system is to classify the name of the move a dancer is going. In this case, the feature algorithm generates a skeletal model with approximate 2D joint locations. The features can be dimensionally lowered by taking the angle of each joint at its vertex, lowering the total dimensions by one. In this case a K-Nearest neighbor classification algorithm might not be enough to classify the action, rather a more complicated probabilistic model such as Hidden Markov Model (HMM) might be needed to classify many action classes correctly.

The classification process always needs to be trained to identify the different classes of possible actions. Actions are highly variable in structure, so no two performances of the same action by the same person are the same. The classifier has to have the ability to classify actions performed by different subjects of different size and gender, as well take into account the variability of the performance in speed and style. It is always a challenge to design an action model that is highly adaptable to all forms of variation. Weinland et al. [27] provides a good representation of the action learning/classification data flow process as shown in figure 2.7. In Weinland’s model, the four key components work together to classify an action. The feature extraction extracts the required features which are fed into the action based components. The action classification component uses the action model database, which was generated using the action learning component, to classify the right action.

Recognition can be temporal based, or static, i.e. from a single frame such as Cheema et al. work in [4]. That means, each frame is independent from the previous frame. Most approaches are temporal based, however, it adds a dimension to work with, making the process slightly more complicated. However, most complex actions can not be classified using only one frame.

Actions can be simple gestures such as a hand wave, leg raise etc. Actions can also be more complicated primitive actions such as walking, sitting down, etc. Simple gestures are easier to classify than more complicated ones. However it is possible to classify a primitive action using a group of gestures. For example, an action ”walking” would involve the gestures, leg up, leg down, standing upright, etc. In the same way, primitive actions can be grouped to identify a
2.3 Mid level: Human Activity Recognition Systems

The mid level Recognition system is designed to integrate with the low core technology by using the extracted results of the low core system. Based on Shian-Ru et al., there are three typical types of [15] human activity recognition systems:

- Single person activity recognition
- Multiple people interaction
- Abnormal activity recognition

Simply put, single person focuses on only one person, and could have a detailed recognition map. The multiple people focuses on people interactions, and used to recognize specific actions, including crowd behavior, and is used mainly in public surveillance. Lastly, the abnormal recognition is used to recognize unfamiliar actions (unfamiliar to the dataset), actions that do not fit into the normal action category, and has many possible applications, including being integrated into either of the mentioned above.

Figure 2.7: Action learning and classification [15]

more complex action such as kicking a ball, cooking, etc.
In a system such as the health monitoring system, the single person activity recognition system would work best, since the algorithm is usually looking for known "warning" actions that could trigger first response.

2.4 High level: Applications

Lastly, the high level, applications module, is what determines the application base and transforms the result of the previous two modules into usable form. In a health monitoring system, that could translate into a phone call, or an alarm system signaling an abnormality in a person’s well being. Other forms of system exist, such as the entertainment system, and is what the Kinect camera was originally built for. Also, one of the highly required systems in the market is the surveillance system, which can be used, for example, as a quick warning system, or to aid staff members detect points of interest faster.

2.5 Motivation for the proposed approach

With an overall view of the system, in this thesis we will focus on the hardware and data fusion part of the home health monitoring system. With that in mind, we explore other related work to gain a better understanding of what is needed. The research will be presented in a logical order leading to the choice of design.

2.5.1 Single camera feature extraction and action recognition

The most basic hardware setup for action recognition, not necessarily the least complex system though, is through the use of a single camera for action recognition. The setup usually involves the use of one camera for the setup. The effectiveness of this approach greatly depends on the combination of proper segmentation, right feature representation, and the correct action recognition algorithm.
In the first example, Cheema et al. [4] demonstrated the use of binary silhouettes to extract key poses for action learning and recognition. The approach is designed with real time performance in mind, so the feature representation is a simple contour based pose representation. They start by finding the center of the extracted binary silhouette, and then from the center, measure a predetermined number of line lengths reaching the contour of the silhouette, effectively transforming the binary data into a distance space, as shown in figure 2.8. Key poses are recognized through the use of the K-means clustering classifier, and the pre-compiled MuHAVI data set (The MuHAVI data set is a public data set developed specifically for action recognition [26]). Based on the authors testing results, the test achieved good results at an average of 56fps using a regular laptop computer running matlab.

In another example, Dalal and Triggs [8] demonstrate the use of Histogram of Oriented Gradients (HOG) as a feature set for human detection. Compared to the previous example, the algorithm is built to detect the human "object" rather than recognize actions. However, recognition is still needed to detect the present of the target object. They rely on the linear Support Vector Machines (SVM) classifier for object detection.

More examples include Gehrig et al.’s [11] work where they demonstrate the use of optical flow motion gradient histograms. Yilmaz and Shah [31] build 3D volumes to describe actions
by exploiting people contour-point tracking. These related works, however, only attempt to address single camera scenarios. Many actions are hard to recognize with single views, and generally are not capable of handling occlusions. The shape and motion information vary greatly to represent a specific action efficiently, since this information is highly dependent of viewpoint.

2.5.2 Stereo camera setup

A multicamera setup aims at improving results in the action recognition field, especially where occlusions are present. However, this setup lacks any spatial data, and relies on 2D data. Different authors have attempted using stereo cameras to acquire depth data. Harville and Li [13] and Darrell et al. [9] used a stereo camera in their approach to extract the depth information in the attempt to better track multiple people in a frame. Both papers use the depth information to better segment the foreground from the background data, and extract silhouettes to work with. Although Cheung and Woo [6] have successfully attempted to deal with occlusions in a stereo camera setup, they are generally not as robust when it comes to occlusions as a multi view camera setup. Extracting 3D data is possible in a multi view setup, however computation cost is much higher than with a stereo camera setup, and might not be practical in real world applications.

2.5.3 Multi view camera setup

In a multi view camera system, each camera has a slightly different view of the same scene. Mustafa el al. [1] developed a method to handle object (humans and other objects) tracking even in the presence of occlusions by utilizing a multicamera setup. In the event a tracked object gets heavily occluded from one camera, the system can keep track of the object by using the other camera where the object is still visible. Furthermore, Mustafa el al. improves occlusion handling by exploiting geometric and dynamic constrains between the two cameras, the trajectory of the target, captured from different view points, can be predicted from the tracking
data. A similar approach has been demonstrated by Calderara el al. Instead of tracking an object as a whole, Calderara el al. aims at tracking a certain number of automatically segmented relevant areas of the human silhouette, which describe the motion for action recognition [2]. Their proposed approach employs the use of a multicamera setup, with the use of Mixture of Guassians for segmentation.

### 2.6 Microsoft Kinect

The Microsoft Kinect camera (Kinect will be used interchangeably with Microsoft Kinect throughout this thesis), is a Microsoft product originally built for the purpose of in game gesture recognition at an affordable price. However, developers quickly realized its potential, and its capabilities that could be matched to other state of the art depth sensing cameras, but with a much lower price tag. Kinect achieves comparable results to continuous wave amplitude Time of Flight (TOF) cameras when trying to extract depth information [17]. Many publications have already been made under the computer vision banner. Xia et al. developed a method to detect a human from different poses by using depth information from Kinect [29]. In a different publication by Xia et al., the authors attempt to use the method developed by Shotton et al. [25] to extract the 3D skeletal joint locations from Kinect’s depth image, and uses histogram of 3D joint location as features for their action recognition algorithm [30]. Kinect’s complementary nature of depth, and color data opens up many opportunities to solve problems in computer vision [12].

With Kinect’s skeletal tracking through Shotton el al.’s method [25], developers can focus on the application and leave the pose recognition hard work to the Kinect. Kinect showed promising results in health care applications. In a controlled body pose, Kinect’s joint estimation is comparable to marker based motion capture, making it a low cost alternative to similar rehabilitation based equipment. Huang et al. attempted to use the Kinect in their Kinerehab system with promising results [3]. Roy et al. [24] also attempted a similar approach using
Kinect in a very low cost system with good results.

Kinect however is still a single device, and its ability to handle occlusions is not very effective. Obdrzalek et al. [22] tested the reliability and accuracy of the Kinect’s human pose estimation based on Shotton et al.’s method. Obdrzalek et al. attempt to compare the Kinect pose estimation to more established pose estimation techniques used in motion capture. The depth accuracy of the Kinect depth sensor ranges from 1-4cm at a range of 1-4m. Their results conclude that the Kinect pose estimation fails in the presence of occlusions, even self occlusion of other limbs or facing away from the camera can result in inaccurate inferred results. However, when fully facing the camera, the Kinect achieves results comparable to motion capture. Since the Kinect was originally build for the Xbox 360 gaming console, it was assumed that the user would constantly be facing the camera. See appendix A for Kinect hardware specifications.

2.7 Multi view Kinect setup

In an effort to address some of the said drawbacks, we came up with a novel approach where we utilize the strength of the Kinect in a multicamera setup. By using the multi view capabilities of a multicamera, the setup is able to deal with occlusions better, and through the Kinect, depth and skeletal data can be extracted. In addition, we propose a way to project the skeletal data from the Kinect to the secondary camera in the setup. This provides additional capabilities which can be utilized in different scenarios. The use of a dual Kinect setup has been considered, however, the infrared dot projections used by the Kinect’s Infra Red (IR) emitter for depth measurement can interfere with the second Kinect’s IR emitter, resulting in undesired results [21].
2.8 Calibration

2.8.1 Brief overview

Camera calibration is an important part in computer vision, and is used extensively in many applications, especially in distortion correction applications, also called geometric camera calibration. The aim of this calibration is to estimate the parameters of pinhole camera model, such as the one shown in figure 2.9. An ideal pinhole camera model consists of a small aperture (pinhole) that lets light into a light proof box that inverts the image and projects it on the opposite side of the box. A camera is a non-ideal pinhole camera due to the presence of a lens, which causes distortions in the projected image. Through the use of calibration, we can estimate the pinhole camera parameters, represented as a $3 \times 3$ matrix called the camera matrix or the matrix of intrinsic parameters, as shown in equation (2.1). The matrix of intrinsic parameters is an estimation of the camera’s internal properties, and these values are fixed as long as the same camera is used, and no zoom applied. Therefore, it is sufficient to estimate these parameters once.

![Figure 2.9: Pin hole camera](image-url)
Chapter 2. Literature review

\[ A = \begin{bmatrix} f_x & \gamma & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \]  

(2.1)

Where:

- \((c_x, c_y)\) is the principal point at the image center
- \((f_x, f_y)\) are the focal length
- \(\gamma\) is the skew of both axis in an image. Figure 2.10.

Figure 2.10: Pin hole camera Model (Excluding the skew)

A second matrix that the calibration generates is the matrix of extrinsic parameters as the one shown in equation (2.2), also known as the transformation matrix. This matrix describes the camera’s motion around a static scene, usually the calibration object. The first 3x3 side of extrinsic matrix defines the rotation, and the other 3x1 part defines the translation. Both the intrinsic and extrinsic matrices are used to estimate the distortion coefficients to correct the image distortion caused by the camera lens.
2.8. **Calibration**

\[
[R|T] = \begin{bmatrix}
  r_{11} & r_{12} & r_{13} & t_1 \\
  r_{21} & r_{22} & r_{23} & t_2 \\
  r_{31} & r_{32} & r_{33} & t_3
\end{bmatrix}
\] (2.2)

### 2.8.2 Single camera Calibration

Calibration is achieved by comparing a calibration object whose 3D geometrical pattern is precisely known. In our approach, we will be using the calibration technique developed by Zhang [32]. Zhang’s proposed technique is easier to implement than other known techniques such as *Three-dimensional reference object-based calibration*, and is more reliable and accurate than *self-calibration* techniques.

Zhang’s method requires a camera to observe a planar pattern such as the captured chessboard pattern shown in figure 2.11b in two different orientation and compare it to the calibration object chessboard pattern with an exact known size as shown in figure 2.11a.

A vector is extracted from the inner boxes of the chessboard pattern in the captured image as well as the calibration object chessboard, such as the ones shown in figure 2.11d and 2.11c respectively. The image vector content is composed of 2D points, while the object vector content is composed of 3D points with the Z component assumed as zero since the chessboard is used as a plane. A 2D point is related to a 3D point by the equation

\[
s\tilde{m} = A[R|t]\tilde{M}
\] (2.3)

where:

- s is an arbitrary constant
- \(\tilde{m}\) is an augmented 2D point to get homogeneous coordinates
- A is the matrix of intrinsic parameters
- \([R|t]\) is the matrix of extrinsic parameters
• \( \tilde{M} \) is an augmented 3D point to get homogeneous coordinates

Based on the assumption of a zero Z component, and without going into too much detail, Zhang’s method computes both the intrinsic and extrinsic parameters of a camera. In our approach, we use OpenCV (Intel® open source computer vision library). OpenCV implements its calibrations based on Zhang’s method.
Chapter 3

Method Architecture and Justification

In this chapter, we present the architecture of the work in this thesis, justify the use of different methods, and explain the approach used in different components. The aim of our approach is to provide a basis for a more flexible pose estimation, action recognition and other computer vision applications. In our approach, we:

- Set up the depth sensing Microsoft Kinect camera and a non-depth camera in a multi-view setup
- Use calibration methods to find both intrinsic and extrinsic matrix parameters of each camera
- Use the newly found intrinsic and extrinsic parameters to perform stereo calibration, and find the transformation matrix between the two cameras
- Extract the translation and Euler rotation matrices from the transformation matrix
- Use Shotton et al.’s [25] algorithm to estimate the human pose from the Kinect’s depth sensor data
- Use Project the Skeletal view from the Kinect’s image plane to the non-depth camera’s image plane using the found translation and Euler rotation matrices
• Fix calibration errors causing projection mismatch

  Use Kinect to find new Euler rotation matrices

  Use feedback iteration to find new translation matrix

3.1 Setup overview

3.1.1 Brief Introduction

Unlike a conventional camera setup, a multi-camera, or in this case, dual-camera setup, must know the physical constraints of the other camera to effectively be able to harness the advantages of a dual-camera setup. The aim of this setup is more health care oriented, so it should have the available properties to work in home environment with lots of furniture and possible occlusions. With that in mind, the system does not need a dynamic position monitor. Since the position of the cameras is going be fixed at the time of use, the location needs only be determined once, and updated only when the position of cameras need to be changed. However, having some kind of error correction in case a camera is accidentally moved slightly from position would make things simpler to handle.

In any multi-camera setup, one camera has to be the center of reference. To make things easier, we will be using the Kinect camera as the center. As the center, the world axis origin would start from the Kinect camera, and the other camera’s physical constraints will provide relative distance between them.

The goal of this setup is to extend the Kinect’s depth sensor capabilities to the other camera, while at the same time, provide the ability to exploiting the advantages that come with a multi camera setup. However, for this to work effectively, both camera’s plane of view must partially overlap (see section 3.1.2 for details). By sharing the depth sensing capabilities of the Kinect with the second camera, we can achieve better pose estimation than what would be achieved with a 2D RGB camera.
3.1.2 Design

Unlike a single camera setup, spatial location of the cameras is a critical parameter that must be known. In a typical 3D world space, three degrees of freedom (DOF) are sufficient to know the exact location of the camera. However, this is not enough to define the direction that the camera lens is facing. An object's orientation in 3D space can be accurately defined using 3 dimensional Euler angles as shown in figure 3.1. This totals up to 6 DOF, and since we have two cameras, the total comes down to 12. However, working with 12 DOF in a highly variable space would make thing very complicated and resource intensive. For that reason, the number of DOF's have to be lowered down.

If we need to find the two cameras positions in the real world coordinate axis, then 12 DOF’s are needed. However, we are only interested in the relative distance between the two cameras if we are going to share the depth data from the Kinect. In that case, by using the Kinect camera as the center of the world axis, with the lens direction corresponding to one of the three axis directions, then that would lower down the DOF’s to 6 instead of 12, making the design a lot less complicated. The Kinect camera has an already built in local coordinate axis, through the use of the Microsoft Kinect SDK that is, and we will align that local axis with the
world axis to keep design simple. Figure 3.2 shows the axis layout of the Kinect camera.

![Figure 3.2: Kinect axis directions and Euler angle orientation. The Y axis is towards the top of the camera, the X axis is to the left, and the Z axis is in the Kinect camera lens direction](image)

Since the cameras are fixed in their location, that generally means that they will be put on a flat surface. So practically, the cameras lens forward direction will be perfectly aligned with the horizontal plane (the plane parallel with the ground), and the lens’s left perpendicular side axis aligned with the x axis. In other words, the camera will have a zero $R_x$ and $R_z$ Euler angle component, the Pitch and Roll in figure 3.1 respectively. This might be true with regards to $R_z$ Roll, however, it was found that many cameras have a slight $R_x$ pitch with respect to the ground parallel horizontal plane by design, and even a small 5 degree inaccuracy could have negative effect on the end result. For that reason, to simplify the problem, only the $R_z$ Euler component will be considered to be zero. This leaves us with 5 DOF to work with, which is much better to deal with than 12 DOFs.

For the setup to work where the Kinect is able to share the depth information with the camera, both cameras must have partial view sharing as shown in figure 3.3. There is no restriction on the direction and angle of the two cameras or the position that they are setup with.
as long as the view from both cameras partially overlaps. The setup is build to work, even if one camera is higher from the ground than the other camera. Fixing the Kinect local axis with the world axis does not inhibit the Kinect from being moved. Rather, if the Kinect is moved, the world axis will be re-established at the Kinect’s local axis location.

Figure 3.3: A Dual Camera setup between the Kinect and a standard camera. The light triangle signifies the Kinect’s view area, and the darker signifies the standard camera’s view area. Both triangles intersect in the view overlap section

So far, this concludes the hardware setup. The next step is to build the software side for the setup to be of any use. The two cameras have to be aware of each others location and angle difference, and doing this manually, for example by using a measuring tape, is a tedious and error prone method. Since accuracy is critical, this is not an option. An accurate way to measure distance and angle difference between the two cameras can be achieved by using calibration techniques, more details can be found in section 2.8.

Upon a successful calibration, the next step is to use the Kinect’s depth data to estimate the pose of the human body and build a skeletal structure such as the one shown in figure 3.4. Since the depth data from the Kinect is shared with the other camera, it is possible to project the constructed skeletal structure into the other cameras image plane. With this, it is possible
to construct two skeletal structures from two different views, but using only the resources to construct one, simplifying the algorithm to achieve the required real time performance.

We also will be introducing a new method to correct calibration errors, or accidental camera movement which might change the captured physical data. Figure 3.5 summarizes our main contribution and shows the process outline in more detail. The figure is also an outline of the logic flow of the software design. Check section 3.5 for more details.

In figure 3.5 the process starts with calibration of each camera individually (*Kinect and camera calibration*) to find the intrinsic parameters, and feed this information into stereo calibration process (*Stereo Calibration*) to find the relative transformation data between the two cameras. The transformation data is then stored (*Distance and Angle database*) and used by the projection process (*Kinect Skeletal projection to camera box*). In the main process loop, the pose of each frame of the *Kinect depth stream* is estimated (*Kinect skeletal pose estimation and construction*) which is then projected onto the second camera image frame (*kinect Skeletal projection to camera*). The main loop can invoke an interrupt (*Does projected pose match actual body*). This could be either manual or automatic where the matching criteria is invoked every
number of image and depth frames processed (We only cover the manual process in this thesis). If the interrupt results in a no match scenario, the main process loop stops and the calibration refinement process starts. This process extracts the body contour (body contour extraction), finds the center point of the extracted contour (center point extraction), estimates the new angle and distance values (angle correction and distance correction) and updates the database (Distance and Angle Database). The process then flows back into the main process loop where it resumes the pose estimation and projection processes until a user ends the process.

The next sections in this chapter will cover the following:

- Stereo calibration to get the relative distance and angle between each other

- Estimation of skeletal pose from depth data using Shotton et al.’s method in the paper "Real-Time Pose Recognition in Parts from Single Depth Image" [25]
Projection extracted skeletal data to the other camera’s image plane

Error correction by segmenting body from the camera’s RGB color data and updating the new geometric data

3.2 Stereo-calibration

A single camera Calibration yields two matrices, the matrix of intrinsic parameters and the matrix of extrinsic parameters. The extrinsic parameter is the relation between the camera and in this instance, the calibration object. Knowing this, if both cameras were calibrated with the same chessboard pattern at the same time, as shown in figure 3.6, then the local axis position of each camera will be relative to the same arbitrary world axis. In this case, we have three point locations, camera 1, Kinect, and the world axis origin, and two rotation matrices. In our approach, we use OpenCV’s stereo calibration algorithm that uses these parameters to extract the relative parameters between the two cameras.

![Figure 3.6: Stereo Calibration step, the two pictures were taken at the same time](image)

The stereo calibration method produces a transformation matrix that relates the position of the second camera to the Kinect camera such as the one shown in equation (2.2). This matrix can be split into two matrices, equation (3.1) and (3.2a). The translation matrix in equation
(3.1) is a vector of the form $t_xi + t_yj + t_zk$ which defines the vector distance between the two cameras’ axis. The rotation matrix in equation (3.2a) is a generalized matrix as the one shown in equation (3.2b). Each of the $x, y, z$ component of the equation in (3.2b) is a matrix shown in equation (3.3a), (3.3b) and (3.3c) respectively. These matrices are then used in the skeletal projection part of this paper, section 3.3.3.

\[
T = \begin{bmatrix} t_x & t_y & t_z \end{bmatrix}
\]  

(3.1)

\[
R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}
\]

(3.2a)

\[
R = R_x(\gamma)R_y(\beta)R_z(\alpha)
\]

(3.2b)
Chapter 3. Method Architecture and Justification

3.3 Pose estimation

In this section, we will present how the depth information is extracted, how a pose is estimated from the depth information, and how this information is shared with the other camera.

3.3.1 Depth extraction

We covered in section 2.6 about the Kinect camera and its built in depth sensor. We will be using Microsoft API build specifically for use with the Kinect to extract the depth information.
3.3.2 Skeletal extraction

Our method is based off of Shotton et al. in [25]. The method uses the advantages of depth over traditional intensity images by eliminating the need to deal with texture problems. This simplifies the background subtraction needed, which is usually a lot less reliable, and takes more processor power to deal with normal intensity images. The method then splits the body depth silhouette posture into a number of color coded labels such as the one shown in figure 3.7.

The labeled data is then classified using a trained classifier which works by matching against a database of a wide array of captured poses. Motion capture equipment is used to capture these poses in almost absolute accuracy to build 500K frames of different poses. However, since the body is capable of a much wider degree of poses, a semi-local body part classifier is used to generalize unseen and relatively close poses. The classifier is also built to handle static poses, and not temporal based motion data, which aims to simplify the algorithm for real-time performance with home equipment. The aim is to produce results similar to the model in figure 3.4.
3.3.3 **Skeletal Projection**

Before, we talked about depth information sharing between the Kinect and the camera. To do this, we have to transform the Kinect depth view plane to the camera view plane so that the depth values are with respect to the camera. Figure 3.8 gives a top view to serve a better understanding.

![Figure 3.8: Kinect and camera in world axis. The figure shows the equivalent depth value of Kinect with respect to the other camera’s local axis. The Kinect is capable of depth values up to 3.5m, however, the other camera is not restricted with this value.](image)

For example, say point P with distance \((x,y,z)\) from the world axis is a depth pixel in the Kinect plane, and we want to find its equivalent depth value from the webcam’s perspective. To do this, we will need to find \(d'_x\), \(d'_y\) and \(d'_z\) as shown in figure 3.8, where \(d'_y\) (not shown in figure 3.8 since \(d'_y\) is perpendicular to the figure’s view plane) is the Y axis distance with respect to the Point P. The easiest way to do this is to transform the points respective world axis to the camera’s local axis using the calibration generated transformation matrix. The first step is to translate the point with matrix (3.1), and then rotate it using \(R_x(\gamma)\), \(R_y(\beta)\) and \(R_z(\alpha)\) from matrix...
(3.3a), (3.3b) and (3.3c) respectively. However, since we are assuming $\alpha$ in $R_z$ is zero, as was discussed in section 3.1.2, we will not be using $R_z$ in the rotation process. This process is then repeated for every depth pixel in the image. The resultant value can be seen in equation, where $N$ is the $N$th depth value and $P'_x$ is the new transformed point value.

$$
\begin{bmatrix}
P'_x(N) \\
P'_y(N) \\
P'_z(N)
\end{bmatrix} =
\begin{bmatrix}
1 & 0 & 0 \\
0 & \cos \gamma & -\sin \gamma \\
0 & \sin \gamma & \cos \gamma
\end{bmatrix}
\begin{bmatrix}
\cos \beta & 0 & \sin \beta \\
0 & 1 & 0 \\
-\sin \beta & 0 & \cos \beta
\end{bmatrix}
\begin{bmatrix}
P_x(N) \\
P_y(N) \\
P_z(N)
\end{bmatrix} -
\begin{bmatrix}
T_x \\
T_y \\
T_z
\end{bmatrix}
$$

(3.4)

However, these is one problem with this setup. The depth sensor has a resolution of 640x480 (Note that the depth resolution is not related to the RGB resolution since each is captured using different sensor components on the camera), which makes a total of almost 300K values to project, which will greatly impact performance. One way to solve this problem would be to instead lower the resolution to 320x240, a total of 76K values, four times smaller than the higher resolution. Even at this value, the performance suffers, and reducing the resolution even further will render the approach not useful.

Our aim in our approach is to produce two pose estimations from two different cameras. So instead of projecting the depth values, we can project the pose estimation itself. Shotton et al. method generates 20 joints and 19 bones connected to these joints as shown in the skeleton model in figure 3.4. 20 joints is by far less than 76K, and projecting 20 should not have any impact on performance. However, this adds an extra step to the problem. Since the Skeleton is extracted in 3D meter coordinates, we will need to apply a 3D to 2D image pixel plane projection.

3D to 2D projection is a computer graphics technique used to transform a 3D model to a 2D image. Since the computer screen is a two dimensional surface, 3D projection is a way to transform 3D geometry that can be rendered as a 2D image [10]. The desired 3D projection type in this thesis is perspective projection, since it preserves the distance, object size relationship, just like the human eye views the world. For more details, see appendix B.
3.4 Calibration refinement

3.4.1 Overview

The whole skeletal projection step is heavily dependent on the success of the stereo calibration process. However, the stereo calibration was designed to handle cameras with a very similar view, that is, they are very close to each other, such as a stereo camera where the two lenses are right next to each other. Any small error in the measurements would produce unreliable results. Also, since the calibration process in our approach is a one time per setup, any small changes to the position of the cameras, say through accidental movement, would produce inaccurate results. To solve this problem, we proposed a feedback system that tries to correct any measurement errors.

Before going into the technical details, let’s explain the logic flow. The method relies on a previous system calibration, and an upper body view of person in both cameras. We will start with Euler orientation correction. If a person was to stand upright facing the Kinect, with both arms to the side with a slight space between the wrists and the hips, such as the one shown in figure 3.10, then the angle at which the user is standing relative the camera can be extracted. Using this in a dual camera setup, the person faces the non-depth camera at a zero angle, and use the Kinect depth to find the angle at which the person is standing. This will lead to finding the horizontal angle $R_Y$ in figure 3.2 between the two cameras. This approach cannot be used to find the angle $R_X$, however, $R_X$ is usually small, and fixing it is not as critical as fixing the other parameters.

The next step is to find the translation error between the two cameras. The body center in the non-depth camera is compared to the projected skeleton’s body center. Based on the compared data, the value is iterated until a match between the two body centers is found. The reason for such an approach, and not a direct method is because we are comparing image pixel values, and we are trying to find actual distance measurements between the two cameras. A reverse projection method, that is by converting between 3D world values, its corresponding
2D image pixel value and vice versa, would not work. The reason for that lies in the fact that we do not have any depth data available from the non-depth camera. This will be explained in more detail in section 3.4.3.

### 3.4.2 Angle correction

As was mentioned before, this step is a correction step, so previous calibration measurement is required. The method builds off of the previous calibration measurements, and tries to correct small changes. The first step is to find the new angle between the two cameras. When a skeletal projection misalignment is noticed by the user, they can invoke a correction parameter in the program. This can be automated, however it’s both resource heavy, and prone to false positives, since the method is designed to work with the person facing the two cameras. On a correction call, the program asks the user to stand facing the non-depth camera, but be in a position where the Kinect camera can see him, as shown in figure 3.9. Both hands must be visible to the Kinect camera, so that gives a limitation to the angle level that can be found.

We need two points in this method, one for the right hand $PR$, and one for left $PL$, and their value should be relative to the Kinect’s world axis. $PR$ and $PL$ each have 3 coordinates, $(x, y, z)$, however, only the $x$ and $z$ coordinates are needed for this method. If we extend a line between the two hands, such as the one shown in figure 3.9, then we can find the angle ($a$) at which the person is facing the Kinect camera using equation (3.5). Since the person is facing perpendicular to the line of sight of the non-depth camera, using triangle rules, we can find angle ($b$), where ($b = 90^\circ - a$) is the relative Euler angle defining $R_Y$ between the Kinect and the non-depth camera.

$$a = \tan^{-1}\left(\frac{PL(z) - PR(z)}{PL(x) - PR(x)}\right)$$  

(3.5)

The next step is to find the needed distance correction.
3.4.3 Distance correction

The distance correction step is a bit more tricky than the previous step. Since depth information is not available in the non-depth camera, we cannot reverse project, that is, use the 2D to 3D projection method. Instead, we will assume a constant depth value, and change the $X$ and $Y$ value offset of the skeletal projection parameters to achieve a match between two reference points, one from the image data of the non-depth camera and one from the Kinect’s skeletal data, such as the one shown in figure 3.10.

To match the projected skeletal information and its corresponding image location, we need to use a point of reference between both data. In the case of the projected skeleton, the hip center point, shown in figure 3.4, is a good center point, since it is not affected by orientation. To get the equivalent point in the image, we will need to find the hip center point as well. This
point can be extracted in a number of ways, including body segmentation using image processing techniques, silhouette dissection, etc. This will be explained in more detail in section 3.4.4. Figure 3.10 demonstrates the distance correction approach.

The next step is to distance match the two results. There are two requirement, first, the distance is estimated to determine if it falls within a certain threshold, and second, direction in which the skeleton has to move needs to be found. Based on the established direction and if the distance value is more than the threshold limit, the skeleton is moved in the 3D world coordinates using a predetermined default distance, usually 2 cm or so, both in the X and the Y direction. The moved skeleton is then re-projected onto the 2D image plane as shown in figure 3.11, and the process repeated again. Should the distance fall within the threshold distance limit, the algorithm is stopped, and the recalibration is complete.
3.4.4 Center Point extraction

In this step, we demonstrate the method used to extract the hip center of a wide stance position posture of person in an image. The approach is simplistic in design, and relies on geometric locations rather than a recognition based system. Theoretically, the method can produce accurate results if the following criteria are met:

- The person is forward facing the camera
- A small part of the legs have to be shown

The first step in the method is to isolate the human silhouette from the background using background subtraction. Without going into too much detail, and using one of the approaches described in section 2.2.1, we will assume that the silhouette is available such as the one shown in figure 3.12a. We can easily find the center of the silhouette $S_C$ by averaging the pixel locations in the binary data. However, the extracted point can not be used as the main center
point for the calibration refinement process. In our design, our approach is meant to work even when part of the legs was cut off from the image such the one shown in figure 3.12e. So, in a scene with the legs not visible, the center point would not be the same as when the body silhouette is fully visible, causing the results of the process to be highly variable with each iteration, as is clearly shown in figure 3.12c and 3.12f.

The point of our approach is to produce a reasonably robust system capable of adapting to such situations. Based on Shotton et al.’s skeletal model, the center point is the HIP_CENTER joint as the one shown in figure 3.4. The point lies somewhere between the Axilla (Arm pits) and the lower hips as shown in figure 3.12d and 3.12d. The center point has to be the same, regardless, to a reasonable degree, of limb visibility. To achieve this, we need to be able to find the Axilla and the hip joints on the silhouette.

![Silhouettes and edges](image)

Figure 3.12: Silhouettes and edges
To find the center point, we need to find four points, A, B, C and D labeled on figure 3.12d. After finding the center weight of the silhouette $S C$, the silhouette’s contour is extracted. This can be done either using elastic Snake contours such as the ones used by Kass et al. in [14], or an edge detection algorithm. The latter being easier to use, we adopt that approach. Several edge detection algorithms are plausible, such as Sobel, Prewitt or Laplacian of Gaussian (LoG). LoG, having produced the best visual result, is used. We then use Dijkstra’s shortest path algorithm to extract a vector $V_{[leng]}$ of points around the extracted edge from the top point, around the silhouette to the same point again. The distance between the weight center and each vector point is calculated, and recorded in a new vector $V_d$. We then plot the vector $V_d$ against the vector’s position number to get a graph such as the one shown in figure 3.13a. The vector value is smoothed using a Gaussian filter to get the graph in figure 3.13b.

Figure 3.13: Silhouettes and edges
The idea of the approach is to find the local minimum from the graph that corresponds to the Axilla points A and D, and the center local minimum that corresponds to the groin area, which is used to find points B and C. However, as can be seen in plot 3.13b, points A and D do not achieve a local minimum. Instead, we increase the height of the center point \( SC \) using equation (3.6), where \( SC' \) is the new weight center point and \( V[1] \) is the first value of the extracted contour which corresponds to the top point of the head. In other words, we are finding half the vertical (hence the Y component in the equation) distance of the current center point \( SC \) and the top head point \( V[1] \), and adding it to \( SC \) to find the new center point \( SC' \), shown in figures 3.12c and 3.12f. This step makes sure that both Axilla points achieve a local minimum on the graph, as can be shown in graph 3.13c. The approach also works well where legs are not fully visible as shown in graph 3.13d.

\[
SC'_y = SC_y + \frac{V[1]_y - SC_y}{2} \quad (3.6)
\]

In the final step, points A and D are found using the local minimum, and points B and C are found by finding the first two points that interest the horizontal line passing to the right and left of the groin local minimum. Using these four points, the image HIP_CENTER equivalent point can be found by finding the intersection between the lines \( \overline{AC} \) and \( \overline{BD} \) as shown in figures 3.12d and 3.12h.

3.5 Software

This section covers the software side of the implementation of the thesis. We use three libraries in addition to the C++ 11 standard library, the OpenCV version 2.4.6 library, Microsoft’s Kinect Software Development Kit (Kinect SDK) Version 1.7, and the Windows C++ library. The OpenCV library supplies a vast array of optimized image processing tools aimed at producing better results and with greater performance and efficiency than what would be achieved should the tool been developed from scratch. The Kinect SDK supplies the tools necessary, in-
including the Kinect driver, to interface with the Kinect camera and extract the depth and image data, in addition to many useful tools. The Windows code is used mainly in the use of handlers for the Kinect SDK.

The code logic flow will be explain through the use of UML class diagrams and related back to the flowchart introduced in figure 3.5 to better understand how the different components of the contribution fit on the software side. To make the different components of figure 3.5 easier to reference, and for better viewing, we have included the same flowchart but with alphanumeric labels, shown in figure 3.14. The structure of the UML class diagram will be explained in parts, and the complete system is shown in figure 3.21 at the end of software section. As to make the explanation easier to understand, the figure 3.5 label references will be italicized, class names will be bold, and class operations underlined Courier font.

The program starts with the Main process loop, assuming that the database F4 contains previous transformation recordings (For the purpose of readability, whenever a reference is made between F1 and F12, refer back to figure 3.14). The job of the Main class is to initialize the other classes, display the video from both cameras, map the estimated Kinect pose and the
Figure 3.15: Main_1 class

The Main_1 class uses the camera class to control the camera and start the video stream F1. As for the Kinect, Main_1 uses the class kinectCam to start both video F1 and depth stream F5. The Kinect pose estimation F6 is performed by getSkeletalData(). Figure 3.16 shows the camera and kinectCam classes.

The pose projection process F7 is performed by the skeletalProjection class shown in figure 3.17. The projectData(...) operation performs the pose projection calculation based on the stored transformation data F4 set by setAngle(...) and setCamCoordinates(...).

The calibration F2 and stereo calibration F3 processes are user invoked through the Main_1
Figure 3.17: `skeletalProjection` class

`skeletalProjection`

- `xAngle`: double
- `yAngle`: double
- `camCoordinates`: Point3d
- `projectData`: skeletalProjection(Point3d): void
- `setAngle(xAngle: double, yAngle: double): void`
- `setCamCoordinates(cCamCoordinates: Point3d): void`

Figure 3.18: `calibCamera` and `stereoCalib` classes

`calibCamera`

- `BOARD_SIZE`: Size
- `SQUARE_LENGTH`: double
- `SAMPLE_COUNT`: Size
- `objectPoints`: vector<vector<Point3d>>
- `imagePoints`: vector<vector<Point3d>>
- `rotationVector`: vector
- `translationVector`: vector
- `cameraMatrix`: Mat
- `distCoeffs`: Mat
- `mn`: double

- `create()> `calibCamera(imageSize: int)
- `destroy()> `calibCamera()
- `findChessboard(image: Mat, saveCorresponds: bool, imageSize: int): bool`
- `runCalibration(): double`
- `displayExtrinsic(image: Mat, int): void`
- `displayMatrix(matrix: Mat): void`

`stereoCalib`

- `webcam: camera`
- `kne: kinectCam`
- `cal1: calibCamera`
- `cal2: calibCamera`
- `rotationMatrix`: Mat
- `translationVector`: Mat
- `ruleAngles`: Point3d
- `mn`: double

- `create()> `stereoCalib(webcam: camera, kne: kinectCam, sampleCount: int)
- `destroy()> `stereoCalib()
- `startCapture(): bool`
- `saveCorresponds: bool`
- `startCalibration(): void`
- `displayInformation(): void`
- `getRuleAngles(): void`
- `showChessboardCorrespondsAll(): void`

The calibration and stereo calibration processes are performed using the `calibCamera` and `stereoCalib` classes shown in figure 3.18. This information is stored onto the database using the `Main_1` operation `updateData(...)` which stores the information onto a text file. During program initialization, the transformation parameters are read from the file using `Main_1` class operation `updatePV()` so as to not require a new stereo calibration each time the program is started.

When a user invokes an interrupt for calibration refinement, the `Main_1` class operation `recalibrateResults()` is used to start the calibration refinement process through the `recalibration` class shown in figure 3.19. The body contour and center point extraction are performed using the `capturePoints(...)` operation, and the angle correction and
distance correct $F12$ are performed using the `startRecalib()` operation. The new transformation parameters are then passed back to `Main_1` class where it updates the database and resumes the pose estimation $F6$ and projection $F7$ processes.

The last class `createButtonLayout` is responsible for creating the buttons layout used by the `Main_1` and managing the mouse click callbacks to some operations defined in `Main_1`. The `createButtonLayout` class structure is shown in figure 3.20.

The full system is shown in figure 3.21.
Figure 3.20: `CreateButtonLayout` class
Chapter 4

Testing and results

In the previous chapters, we described our method in detail and explain the logic flow behind each component. In this chapter, we describe the effectiveness of the proposed approach, find out the proper parameters to use, and discuss the results.

We will start by dividing the system into three sections, and testing each part individually. In the first part, we will be testing out the stereo calibration method to find its strengths and drawbacks. In the second part, we will be testing out the pose estimation of Shotton et al.’s method to find its pose estimation limitations, and our proposed projection of the estimated pose to find how well it works. In the third and last part, we will be exploring the capabilities of the proposed calibration refinement method, and make interpretations of its real world application potential from its results.

4.1 Stereo Calibration

The calibration process is an important part of the system. If the calibration is not accurate, then the results are not accurate. So gaining an understanding of the performance and limitations of the calibration process is important in order to be able to determine the cause of errors. Since the stereo calibration process is the first step in the approach, it will make problem isolation and analysis easier if its test data is available before proceeding into the main testing stage. In
this section, three tests will be conducted. Test 1 is a test of result consistency and deviation for the same transformation parameters. In test 2, we will determine what pattern count gives the best results. Finally, test 3 will give an insight into how the process behaves for different camera distances and angles. After each test, we will be discussing the result, any visible shortcomings, and possible problems faced during testing. Since we are only interested in the location data of the two cameras, we will not be looking at the intrinsic results, we will only focus on the 3D translation and Euler orientation.

The hardware used is:

- Microsoft Kinect and Lenovo Camera, both running at a 640 × 480 resolution
- Laptop computer with a quad core 2.4GHz i7 processor and 8GB of Ram

Due to the lack of motion capture equipment or the like, all physical measurements are done manually. This leaves the tests prone to error. The translation uncertainty was estimated to be around ±3cm and angle uncertainty of ±5°

### 4.1.1 Test 1: Pattern recognition

The first test’s goal is to test the consistency of the calibration process to produce the same results when the same calibration conditions and parameters are repeated. We will be testing for both the distance and orientation performance between both cameras. In this test, we will be using:

- $R_x$ and $R_z$ equate to 0, and $R_y = 30^\circ$ of relative angle (The angle representation is shown in figure 3.2)
- A relative Euclidean distance of 41 cm
- 5 pattern views used (That is, the calibration process takes 5 difference views of the calibration pattern into its algorithm)
Test 1 results and discussion

The results are summarized in graph 4.1 and 4.2. The graphs contain the distance and angle results (solid), the mean result (dotted), and the standard deviation (star).

Figure 4.1: Distance consistency performance

For the distance results in figure 4.1, the mean distance value was around 43cm, which is reasonably accurate, considering our estimated uncertainty to be around $\pm 3cm$. The standard deviation value was around 3cm, which most of the test result values fell into, not counting the four outliers. Out of the four outliers, two are truly out of range since the bottom two (run 6 and run 10) are still within reasonable values.

As for the orientation results in figure 4.2, the mean angle value is around $33^\circ$, which is also reasonably accurate, compared to $30^\circ \pm 5^\circ$ actual value. The standard deviation value was found to be around $2.5^\circ$, and most of the test results also fall into.

The test results are mostly acceptable, since the projections process, which will come later,
4.1. Stereo Calibration

Figure 4.2: Angle consistency performance

A tolerance of 5cm and 5° without producing noticeable change. That being said, achieving high accuracy should always be kept a priority, since errors can stack throughout the process flow, and produce undesired results. There does not seem to be any correlation with the inaccuracies of the distance and the angle counterpart of both graphs. This makes it harder to pinpoint the culprit impacting the results, be that a code bug, flaw in the algorithm’s logic flow, human error during the calibration capture process, or a combination of everything. So, assuming the error is mainly human based, can better results be achieved? To answer the question, we need to go through more tests to determine the cause of the errors. This will be discussed at the end of this section after test 3.
4.1.2 Test 2: Measurement accuracy vs pattern count

In this test, we will be testing the extrinsic parameter estimation accuracy of the translation matrix between the two cameras depending on the number of patterns used for recognition. Zhang et al. algorithm [32] was designed to work with at least two different pattern view to estimate the intrinsic and the extrinsic parameters. However, since the authors’ design was used for a single camera, we have to test the approach to find the results for a multi-camera system, and find the best number to achieve the most accurate results. The two cameras will be placed at a known location with a known angle. We will start with two pattern views, and move all the way up to eight views. In this test we will be using:

- $R_X$ and $R_Z$ equate to 0, and $R_Y = 30^\circ$ of relative angle
- A relative Euclidean distance of 55cm
- Each parameter is run 5 times, and the average used in the analysis

Results:

The results of the test is summarized into figures 4.3, 4.4 and 4.5. In the first two graphs, we have the angle and distance data (solid) and the Actual data (dotted). In figure 4.5, the solid line is the distance accuracy data, and dotted line is the angle accuracy data. The results in figure 4.5 are calculated using equation (4.1) where A is the accuracy, C is the current distance or angle value, and V is the actual distance or angle value.

$$A = \frac{|C - V|}{V} \times 100\%$$  \hspace{1cm} (4.1)

In figure 4.3, it is very obvious that the more patterns used, the higher the accuracy of the results. As the pattern count hits seven, the results are very accurate. In figure 4.4, there does not seem to be any form of impact from the pattern count used. In reality, the difference of the measured angle from the actual angle is very insignificant ($1.25^\circ$ at most based on the
4.1. Stereo Calibration

Figure 4.3: Distance performance

Figure 4.4: Angle performance
Figure 4.5: Percentage Accuracy

results), leading to a very accurate test. The results are far more clear in figure 4.5, where the accuracy, using equation (4.1), is represented against pattern count. As can be seen, the distance measurement accuracy rises from 79% to almost 98% when using two to eight patterns respectively. The angle accuracy is consistent and not affected much by the number of pattern counts used, fluctuating between 94% and 97% accuracy. However, to achieve the best results, we will be using the eight pattern count to conduct the remainder of the tests.

4.1.3 Test 3: Measurement Accuracy vs distance and angle between the cameras

In this test, we will be testing the limit of the total distance that the calibration process can detect between the two cameras, and find the maximum angle at which results are still acceptable. Testing the limit of the calibration process provides an opportunity to identify the reasons why the calibration process would fail, which in turn, will define the limits of the overall approach.
Two separate tests will be performed, one testing the maximum distance at zero angle (note that zero angle means the two cameras share the same view axis), and the other testing the maximum angle at constant distance. The parameters used are as follows:

**Maximum Distance (Test 3a)**

- $R_x$, $R_z$ and $R_y$ equate to $0^\circ$ of relative angle
- The X component distance varied starting at 25 cm to a maximum at 5 cm increments
- Each parameter is run 5 times, and the average used in the analysis
- Using 8 calibration pattern views

**Maximum Angle (Test 3b)**

- $R_x$ and $R_z$ are fixed to $0^\circ$. $R_y$ is varied starting from $0^\circ$ to a max possible value of $90^\circ$ at $10^\circ$ increments
- The X component distance is fixed at 50 cm
- Each parameter is run 5 times, and the average used in the analysis
- Using 8 calibration pattern views

**Results:**

The results of the test are shown in figures 4.6a and figure 4.6b. The graphs show the measured distance or angle value (solid) compared to the actual value (dotted). The standard deviation shows the typical variance of the values of each distance or angle value.

In the first test, initially, the distance calculation achieves accurate results as shown by figure 4.6a with a small standard deviation. However, as soon as the distance value hits beyond 60 cm, the results start to lose accuracy and the standard deviation increases. The reason for the inaccuracy cannot be easily interpreted from the graphs. Instead, we will look at the test itself.
Chapter 4. Testing and results

(a) Test3a: Distance

(b) Test3b: Angle

Figure 4.6: Test3 Results
Two things were noticed during the test. First, the chessboard pattern had to be held very still beyond 60cm to get reliable data. Second, there where many outlier values after 60cm, some were completely wrong that they forced the test to be redone. This was especially true at 70cm.

As the two cameras go further apart, the more distant back the chessboard pattern has to be to fit in both cameras views as shown in figure 4.7. The further back the chessboard pattern, the less defined the pattern boxes are. Since we are limited to a resolution of $640 \times 480$, increasing the resolution is not an option and even if it was, the frame rate would drop to an unusable level. So any movement would distort the boxes in the chessboard pattern, leading to the limitation of the calibration process of 80cm at a $0^\circ$ angle. One way to increase the distance measurement limit is by increasing the angle, the reason for which is the increase in camera view overlap.

In the second part of this test, the angle calculation achieves almost perfect results, with a very small standard deviation. However, at 50cm distance between the two cameras, it becomes impossible to perform the calibration process beyond $70^\circ$ as shown in figure 4.8c, since the chessboard pattern can no longer be fully visible in both cameras beyond that angle. Figure 4.8 demonstrates this more informatively, as the angle becomes larger, the view overlap area (location of the robot) becomes smaller, making the bot/calibration object less visible. The
second test did not suffer from the same complications as the first test, since contrary to the first, the increase in angle increases the overlap between the two cameras, to a limit, allowing for a wider view, and hence better performance.

![Figure 4.8: Camera Angle limitation example](image)

**4.1.4 Discussion of test results**

The conclusion of these test is that the calibration process is a success, however with limitations. To answer the question presented in the first test "Can better results be achieved?", we will have to look at the analysis in detail.
In the first test, we demonstrated that the calibration process is reasonably consistent, where the results were almost always the same. In the second test, it showed that higher accuracy can be achieved when more patterns were used, and highest was achieved by using eight patterns. In the third and most important test, it demonstrated that the accuracy of the calibration process is largely dependent on the quality of the captured calibration object. This means, good lighting conditions, reduced motion blur, and larger chessboard box patterns.

The first two can be easily handled, with a good light source, and a calibration object holding stand. As for the chessboard box size, it is limited by both resolution, and the distance of the board from the camera. When it comes to capturing the board from two separate camera views, the distance of the board can no longer be controlled, and this presents the first limitation. As for the resolution, increasing it would potentially solve the problem. A reasonable next resolution step up is at $1024 \times 768$ which is 2.5 times the pixel count, however, for two cameras this means five times more resources allocation needed from the current already resource expensive design. Without major code optimization, or hardware change, the design is limited to the current resolution of $640 \times 480$, which presents the second limitation (In commercial products, the design code would be heavily optimized. In addition to that, the producer would make use of hardware acceleration from GPU chips, which are designed to handle heavy mathematical computation, allowing for possible resolutions as high as $1920 \times 1080$).

When an image is slightly blurred, the chessboard pattern gets misclassified as shown in figure 4.9b, compared to a properly classified pattern in figure 4.9a. Lastly, the Kinect’s produced RGB image is slightly grainy, which also adds to the problem at longer distances (Note: The image on the left is from the webcam, and the image on the right is from the Kinect). So, is it possible to get better results? Yes, but within limits.
4.2 Pose estimation and projection

In this section we will be testing the pose estimation, skeletal extraction and projection between both cameras. We will be exploring the capabilities and limitations of the proposed approach. To make sound judgment, we will first need to evaluate the method proposed by Shotton et al. (Kinect Method for short) to estimate the pose from the depth data of the Kinect camera, since our approach is built on top of this method, and any failure would transfer over. Due to the unavailability of proper motion caption equipment, the test results will be assessed both qualitatively and quantitatively.

4.2.1 Test 4: Kinect Pose estimation

The Kinect camera was originally intended as a gaming device, a type of controller. By extension, the method build to work with it, i.e. Kinect method, was designed to handle scenarios that would most likely occur in a gaming environment. That meant that the method was designed to handle poses of people facing the camera, no occlusions, and the body fully visible. In a real world scenario, some, if not all of these are not always possible. However, the method was built with some form of prediction to infer the location of joints in cases where the joint
location is not obvious.

This test was conducted to test the pose estimation of the Kinect method on the depth data from the Kinect. The test will be varied to test different scenarios, starting with a clear full body view, and ending with some kind of partial occlusion. Any in-test comments will be noted and used to make better analysis of the results.

**Results:**

During testing, we found four different scenarios which summarize the performance of the Kinect method. The scenarios are presented in parts of figure 4.10.

![Kinect Results](image)

(a) Kinect without occlusion  
(b) Kinect with legs occluded  
(c) Kinect with object occlusion  
(d) Kinect with body occlusion

Figure 4.10: Kinect Results
The first image 4.10a represents the scenario where the person is fully facing the camera, no occlusions present, and all limbs shown. Since the pose estimation of the Kinect method was designed to handle this kind of scenario, it functioned almost flawlessly, with real time pose estimation of the body. Similarly, when part of the legs are missing such as the image shown in figure 4.10b, the pose estimation was accurate, showing the partial bone extension of the legs, suggesting that the method is capable of accommodating view plane occlusion.

In the presence of occlusions, however, the method does not work as effectively as before, such as the image shown in figure 4.10c. During the test, and as shown in the image, the method tries to estimate the location of the joints. However, beyond the first joint estimation, which in the case of figure 4.10c is the left knee, it is completely off. Even the left knee estimation is slightly higher than it actually is.

Lastly, the joint estimation fails from self occlusion when the person is not fully facing the camera, such the image in figure 4.10d where the left hand is occluded by the body. However, based in the result in figure 4.10d, it does retain its last known location, and tried to infer its possible position, which to some degree is acceptable.

The Kinect is not a true 3D camera, so for the method to accurately measure position, the joint has to be within the camera’s direct view. This drawback is one of the main reasons for the proposed approach. When a joint is occluded, the depth sensor cannot get a reading on the depth information of that joint, resulting in the method having to estimate the joint’s location.

In figure 4.10b, the method seems to handle the occlusion of a view plane cutoff better than when the occlusion is caused by an object as in figure 4.10c. One possible explanation is that the depth sensor is able to pick up the leg locations, however, the legs projection do not fall onto the 2D clip plane of camera’s view plane, which results in known joint location that is just not visible on the screen. Meanwhile, the object occlusion covers up the leg, preventing the Kinect from picking up accurate measurement. The same concept applies to limbs occluded by the body itself, as can be seen in figure 4.10d. The method is able to estimate the joint locations based on the last seen position, which is good enough in most situations. Lastly,
the method seems to occasionally misinterpret surrounding objects as human poses. However, these usually only last briefly, and are not a major concern. In conclusion, the Kinect pose estimation method is reliable for the most part, however it tends to partially fail where occlusions are present. Nonetheless, the method’s ability to not fully fail during occlusions is a big plus, which guarantees at least some form of result except in the worst of possible scenarios of heavy to complete occlusion.

4.2.2 Test 5: Pose projection

Everything we have tested so far eventually leads up to this test. Our proposed method implies accurate results should the calibration process and Kinect pose estimation perform efficiently. Assuming the calibration method achieves accurate results, we will be going over similar scenarios that were handled by section 4.2. That will include self occlusions from both Kinect and the cameras perspective, as well as object occlusion.

Results:

The majority of the tests can be summarized into four results. Figure 4.11 shows the result of the projection with no occlusions in the way. Figure 4.12 shows the result with the webcam occluded. Figure 4.13 shows the result with the Kinect occluded. Finally, figure 4.14 shows the result with webcam self occlusion.

The first result, shown in figure 4.11, shows the best case scenario of the projection where no occlusions are present, and the human pose is clearly visible from both cameras. This shows that in an ideal situation, the skeletal projection of the extracted pose works very well, given that the calibration method gave accurate results.

In the second result, shown in figure 4.12, the Kinect has no occlusion between itself and the person, and so the pose estimation works without any problem (figure 4.12b). As long as the correct pose is available, the projected pose should function normally. In other words, even if the person was partially occluded in the webcam image (figure 4.12a), the projected pose
will not be affected. Similarly, when the person is self occluding a limb in the webcam (figure 4.14a), the pose estimation is not affected as long as the limb is visible in the Kinect view (figure 4.14b) as shown in figure 4.14

However, any object occluding the Kinect will effect the pose estimation process (figure 4.13b) as was explain in section 4.2.1, and with that, the pose projection will fail at the occluded location (figure 4.13a) as can be seen in figure 4.13. Possible ways to fix this will be discussed in the future work section of chapter 5.
4.3 Calibration refinement

In the last test, we will going through a series of experimentations similar to the calibration process in order to be able to compare the extracted data from the method described in section 3.4. For the purpose of controlling the result’s output as much as possible, we will be selecting the body’s mid point manually rather than relying on background subtracted model that was proposed in section 3.4.4. Just like before, the tests will revolve around finding the calculated distance and orientation data compared to the actual distance and orientation data. The tests
4.3.1 Test 6: Calibration refinement

In section 3.4, we discussed how the calibration refinement step works to find the correct angle and distance parameters of the transformation matrix. The aim of the method is to avoid going through the calibration step again for small changes or minor error correction. The calibration refinement step is sort of an add on to the calibration process. It is designed to use the preexisting calibration values, then working its way to find the proper values. In this test, we will be measuring the difference amount that process can handle. Two tests runs will be performed, one with a constant angle and variable distance, and the other with a constant distance and variable orientation. The parameters used are as follows:

**Maximum Distance (Test 6a)**

- $R_X$, $R_Z$ and $R_Y$ equate to $0^\circ$ of relative angle

- The X component distance varied starting at $30cm$ to a maximum, at $5cm$ increments

- Initial parameters values at distance of $(10,0,0)$ and orientation parameters of $(0,0,0)$

**Maximum Angle (Test 6b)**

- $R_X$ and $R_Z$ are fixed to $0^\circ$. $R_Y$ is varied starting from $0^\circ$ to a max possible value of $90^\circ$ at $10^\circ$ increments

- The X component distance is fixed at $50cm$

- Initial parameters values at distance of $(10,0,0)$ and orientation parameters of $(0,0,0)$
Results:

The results are summarized into graphs of figures 4.15 and 4.16 for the distance and orientation respectively. Figure 4.15a shows the calculated distance compared to the actual distance. Figure 4.15b shows the accuracy of measurement for the distance performance. Figure 4.16a shows the calculated orientation compared to the actual orientation. Lastly, 4.16b shows the accuracy measurement of the orientation performance. Equations (4.2) and (4.3) where used in the accuracy calculations.

\[
\text{avg} = \frac{|d_{\text{measured}} - d_{\text{actual}}|}{d_{\text{actual}}} \times 100\% \quad (4.2)
\]

\[
\text{avg} = \frac{|a_{\text{measured}} - a_{\text{actual}}|}{a_{\text{actual}}} \times 100\% \quad (4.3)
\]

In the first test run, the distance measurement produced result with an average accuracy of 92% (figure 4.15b). That is a good accuracy, however, its not as precise as the calibration process. Unlike the calibration method by Zhang [32] our proposed calibration refinement process accuracy seems to be independent of distance between the cameras, even achieving higher accuracy at longer distances. Previously in test 3, we found that the limit of the distance that can be calculated using the calibration method was around 70cm. Our proposed method was able to double that distance with a value of around 130cm. Beyond that distance, the cameras shared little view overlap, making it impossible to use the calibration refinement method.

The orientation calculation method on the other hand produced better results than its distance counterpart with an average accuracy of 96% (figure 4.16b). The results from test 6b (figure 4.16a) are very comparable to the results from test3b (figure 4.6b). However, the calibration refinement from test 6b was not able to find the initial value accurately compared to the calibration from test3b. Since we are not dealing with a chessboard, the shared view overlap between the two cameras does not have to be as large. This gives test 6b an edge compared to test3b in the maximum possible angle detection. This test was able to achieve a max angle of
Chapter 4. Testing and results

Figure 4.15: Test 6a Results

(a) Test 6a: Distance

(b) Test 6a: Percentage accuracy
4.3. Calibration refinement

(a) Test 6b: Angle

(b) Test 6b: Percentage accuracy

Figure 4.16: Test 6b Results
80 compared to the max angle of 70 in test3b, with both tests having the cameras fixed at 50cm apart.

The calibration refinement method was found to be highly dependent on the person using it. For example, should the hands not be coplanar to the rest of the body, it will throw off the algorithm and produce wrong results. However, it does not take long getting used to using it, and was able to get the provided results from a first-time use.

So, can this calibration refinement step work without any preexisting calibration parameter calculation? At this stage, not likely. While the process does work, it still needs a rough estimation of the depth position between the two cameras. Our method works by assuming the value of the depth is unchanged from the last calibration run, and attempts to find the other two distance components and orientation instead. As a result, should the depth distance component change by a large amount, the method will fail to find the right values. The current method is suitable to be used for correcting the calibration errors and applying small changes, but not as a complete replacement. The next step will be to find a calibration replacement using self-calibration methods as will be discussed in the future work section of this thesis.
Chapter 5

Conclusion and future work

5.1 Summary

In this work, we developed an improved dual camera system that utilizes the depth capabilities of the Microsoft Kinect camera, uses Shotton et al.’s depth pose estimation technique in [25], and projects the estimated pose onto the other camera. Our approach is build to work with action recognition applications, however, the approach can be used in all kinds of computer vision systems.

We have also proposed an approach to re-estimate the transformation matrix to fix any parameter errors results caused by the chessboard pattern recognition. The approach uses the person in the scene of both cameras as the calibration object to determine the proper relative geometric data between the two cameras. The approach extracts the persons mid point by finding the position average of a contour of a background subtracted model. This mid point is used in determining the new relative distance. The position on the hands of the human object are used to find the relative angle.
5.2 Future work

Although the goal of this thesis have been achieved, there still exists a lot of room for improvement. Both cameras are location aware, and this location awareness is used in the projection process. However, much more can be done with this location awareness. One thing that would have been a goal in the thesis should the time have allowed it would be camera feedback. In camera feedback, the non-depth camera in the dual camera pair employs some form of trajectory based limb tracking. The tracking can be used to give the Kinect better pose estimation by using the data from both cameras instead of just one. This is explained in more detail in section 5.2.1.

In addition to the tracking step, the calibration refinement method can be greatly improved to perform fully automated calibration without the need for predetermined transformation parameters. More details will be presented in section 5.2.2.

5.2.1 Improved tracking

The pose estimation method proposed by Shotton et al. [25] for the Kinect’s depth is developed to handle people facing the camera in a clear area. Should the person face sideways, or have a partial occlusion to one of the limbs, the pose estimation fails to capture the said limbs proper joint location. One of the main goals of this thesis was to design a system capable of employing occlusion handling and data fusion. By using two cameras in the pose estimation process, should one of the limbs get partially occluded from the Kinect camera, the other camera can report back the location of the occluded joint. The non-depth camera would have to use some form of human segmentation algorithm, such as the one used by Calderara et al. in [2]. Calderara el al’s proposed system was designed to automatically segment the human silhouette into a number of regions in the image describing the motion evolution. The segmentation is done using a mixture of Gaussians and Expectation minimization algorithm. The segmented body parts can then be linked to the Kinect’s pose estimation model to keep track of any oc-
5.2. Future work

Figure 5.1: Ayazoglu’s el al.’s [1] method in action. As can be seen from the image, the trajectory of the car tracked by one camera is mapped by the other camera with the occluded view.

cluded body parts. This would also greatly improve the pose estimation of a person facing the camera sideways. Ayazoglu et al. in [1] proposed an object trajectory tracking of a target in images, each captured from a different viewpoint. By utilizing this method in our approach, we can keep individual trajectory’s of each limb, and should any limb get occluded, the trajectory can be estimated from the multi view, providing a possible fall back for the Kinect’s Pose estimation algorithm. Figure 5.1 shows more details.

5.2.2 Stereo Self calibration

Self calibration is a form a calibration to extract the intrinsic and extrinsic parameters without the use of a calibration pattern. The method relies on some form of distinct features in the camera image view as the calibration object. That could be any form of straight line or edges, corners, etc. Some proposed methods have used the human pose as a calibration object. Our proposed calibration refinement step that was presented in section 3.4 is not a calibration method per the tradition meaning, since for a method to be considered a calibration, it has to
find the extrinsic and intrinsic parameters of a camera, and our method has the ability to only find five out of six extrinsic parameters, where the sixth parameter is the depth transformation $T_y$, and nothing regarding the intrinsic parameters. However, as was demonstrated by Kuo et al. [16], the calibration of a single camera using the human pose calibration object is possible. Their method employs the use of five coplanar points formed when a person stands in a ”Mid stance” posture as shown in figure 5.2. As with any camera calibration process, it is scale dependent, so the hip width is fixed. The hip width varies from person to person, leading to a rough estimate of the five coplanar 3D model. Their approach involves an iterative method fixing the 3D model, and with it finding the intrinsic and extrinsic calibration parameters.

Tsuhen el al. in [5] introduces an approach to self calibrate and find the relative orientation and distance between two cameras. Their approach relies on the view difference of a human blob calibration object. The straight line produces from the foot and head localization differs
from one camera view to the other as shown in figure 5.3, and the authors’ method exploits that feature to accurately measure the calibrate parameters.

The problem with this approach is that it is designed for a camera system with a relatively distant relative positioning, and our approach is designed for a much shorter distance than Tsuhen el al.’s method. However, it is possible to combine the first and the second approaches to achieve a highly robust system that does not depend on the need for the tradition pattern based calibration system.
Bibliography


Appendix A

Kinect camera

The Microsoft Kinect camera is a depth enabled camera that produces results similar to the Time of Flight (ToF) cameras. However unlike ToF cameras, the Kinect finds the depth through the use of an infrared dot pattern similar to the one shown in figure A.2, projected from its infrared IR emitter, where the dots are not visible to the naked eye. Based on the size of the detected pattern, the Kinect is able to generate a $640 \times 480$ pixel depth map, where each pixel stores a depth value in meters [19] [20] [21].

The camera has a $43^\circ$ vertical by $57^\circ$ view field, with a distance depth detection between 1.2m and 3.5m. Anything outside of the range is declared undefined. The color sensor is capable of a $1280 \times 960$ resolution at 15 frames per second, and $640 \times 480$ at 30 frames per second.

Figure A.1: Kinect camera specs [19]
second. There are four built-in audio sensors. These are used to determine who is talking by locating where the sound is coming from (Mainly used in games where more than one person is in the camera view) [19] [20] [21].
Appendix B

3D to 2D Projection

There are two 3D to 2D projection types, each with its own set of application use. There is the simpler Orthographic (parallel) projection and the Perspective projection types. Simply removing the Z component of a 3D model would not result in a very useful projection, which is why Orthographic or Perspective projection has to be used.

B.1 Orthographic Projection

In any type of projection, there is a canonical view volume, and a view volume. The process involves transforming the view from the view volume to the canonical view volume. The view volume is the region that contains all the necessary geometry that needs to be displayed. The difference between Orthographic and perspective projection is the type of view volume used.

In orthographic projection, both view boxes are axis aligned as shown in figure B.1. This results in no distance correction, and objects of the same size in 3D space will appear the same size in the projection, regardless of depth distance [10] as shown in figure B.2. This is useful in platform graphic game types.

Figure B.1: Orthographic projection view

Figure B.2: Orthographic projection result
B.2 Perspective projection

The second type of projection uses a truncated pyramid instead of a axis aligned box as shown in figure B.3. When transforming from 3D to 2D view, the closer an object is to the far end of the pyramid, the smaller it will appear on the canonical volume. This gives the preservation of distance, similar to the way the human eye, or camera works as shown in figure B.4.


\[4\] http://users.ece.gatech.edu/lanterma/mpg10/mpglecture04f10_3dto2dproj.pdf [18]
Figure B.3: Orthographic projection view

Figure B.4: Orthographic projection view
Curriculum Vitae

Name: Omar Kayal

Post-Secondary Education and Degrees:
2012-2014 M.E.Sc Electrical and Computer Engineering
The University of Western Ontario
London, Ontario, Canada

2007-2011 B.Sc Electrical and Computer Engineering
The University of Western Ontario
London, Ontario, Canada

Related Work Experience:
2012-2013 Teaching Assistance
The University of Western Ontario
London, Ontario, Canada

SE 2203 - Software Design
SE 2250 - Software Construction
SE 3316 - Web Technologies

Honours and Awards:
UWO Graduate Scholarship

Publications: