Gesture Analysis for Human-Computer Interface Using Profile-Matching Stereo Vision

Yung Ping Chang

Brigham Young University - Provo

Follow this and additional works at: https://scholarsarchive.byu.edu/etd

Part of the Electrical and Computer Engineering Commons

BYU ScholarsArchive Citation
https://scholarsarchive.byu.edu/etd/3729

This Thesis is brought to you for free and open access by BYU ScholarsArchive. It has been accepted for inclusion in Theses and Dissertations by an authorized administrator of BYU ScholarsArchive. For more information, please contact scholarsarchive@byu.edu, ellen_amatangelo@byu.edu.
Gesture Analysis for Human-Computer Interface Using Profile-Matching Stereo Vision

Yung-Ping Chang

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Dah-Jye Lee, Chair
Doran K. Wilde
James K. Archibald

Department of Electrical and Computer Engineering
Brigham Young University
July 2013

Copyright © 2013 Yung-Ping Chang
All Rights Reserved
ABSTRACT

Gesture Analysis for Human-Computer Interface Using Profile-Matching Stereo Vision

Yung-Ping Chang
Department of Electrical and Computer Engineering, BYU
Master of Science

This thesis presents a novel profile shape matching stereo vision algorithm. This algorithm is able to obtain 3D information in real time from a pair of stereo images. This algorithm produces the 3D information by matching the profile intensity shapes on the same row of the two images from a stereo image pair. The advantage of this profile shape matching algorithm is that the detection of correspondences relies on intensity profile shape not on intensity values, which subject to lighting variations. The user can choose an interval of disparity, and then an object in a desired distance range can be segmented out from the background. In other words, the algorithm detects the object according to its distance to the cameras.

Based on the resulting 3D information, the movement and gesture of the control agents, in our test cases the human body and fingers, in space in a desired distance range can be determined. The body movement and gestures can then be analyzed for human-computer interface purposes. In this thesis, the algorithm was applied for human pose and hand gesture estimation. To demonstrate its performance the estimation results were interpreted as inputs and sent to a smart phone to control its functions.

While this algorithm does have a trade-off between accuracy and processing speed, we found a balance that can produce the result in real time, and the result has sufficient accuracy for practical use of recognizing human poses and hand gesture. The experimental result shows that the proposed algorithm has higher accuracy and is $1.14 \times$ faster than the original version on tested stereo image pairs.

Keywords: stereo vision, human-computer interface, row profile matching
ACKNOWLEDGMENTS

It is impossible for me to finish my thesis without the guidance of my committees, help from friends, and support from my family and wife.

I would like to express the deepest appreciation to my advisor, Dr. Dah-Jye Lee, who has been my friend, guide and advisor. Without his guidance and help this thesis would not have been possible. I would also like to thank Dr. Doran Wilde and Dr. James Archibald for being willing to be on my committees.

I am most grateful to Beau Tippetts for providing me with the code of the original profile shape matching stereo vision algorithm. The improvement I have made to the algorithm is the result of his help.

As a non-English native speaker, it is hard for me to write a thesis in English. I am indebted to my friends Austin Van Wagoner and Craig Wilson for helping revising my thesis.

I owe my deepest gratitude to my parents who have given me the opportunity to study in the US and supported me.

The most gratitude goes to my wife, Hui-Chun Chan, who quit her job and came to the US to support me. I am grateful for her support and love through these two years.
# TABLE OF CONTENTS

## LIST OF TABLES
- v

## LIST OF FIGURES
- vi

### Chapter 1 INTRODUCTION
- 1

### Chapter 2 BACKGROUND
- 5
  - 2.1 Processing Speed of Real-time Stereo Vision Algorithms
  - 2.2 Real-time Stereo Algorithm for Resource Limited Systems
  - 2.3 Human-Computer Interface

### Chapter 3 PROFILE SHAPE MATCHING ALGORITHM
- 12
  - 3.1 Original Profile Shape Matching Algorithm
  - 3.2 Modified Profile Shape Matching Algorithm

### Chapter 4 HUMAN POSE AND HAND GESTURE ESTIMATION
- 23
  - 4.1 Human Pose Estimation
    - 4.1.1 Haar Face Detection
    - 4.1.2 Human Body Model
  - 4.2 Hand Gesture Estimation
    - 4.2.1 Convex Hull

### Chapter 5 RESULTS
- 38
  - 5.1 Proposed Algorithm Parameters
  - 5.2 Hardware Details and Implementation
  - 5.3 Accuracy Measurement
  - 5.4 Resulting Images of Tsukuba Stereo Image Pair
  - 5.5 Resulting Images of Human Pose Estimation
  - 5.6 Resulting Images of Hand Gesture Estimation

### Chapter 6 CONCLUSION AND FUTURE WORK
- 52
  - 6.1 Conclusion
    - 6.1.1 Contributions
  - 6.2 Future Work

## REFERENCES
- 54
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Comparison of Computational Abilities of Real-time Algorithms</td>
<td>6</td>
</tr>
<tr>
<td>5.1</td>
<td>Parameter list</td>
<td>38</td>
</tr>
<tr>
<td>5.2</td>
<td>Accuracy Comparison of the Original and Modified Profile Shape Matching Algorithm</td>
<td>42</td>
</tr>
</tbody>
</table>
# LIST OF FIGURES

1.1 Microsoft Kinect for Entertainment ............................................. 2  
1.2 Stereo Vision Application for Safe Driving ................................. 2  
2.1 Runtime Comparisons of Real-Time Stereo Algorithms .................. 8  
2.2 Evolution of Human-Computer Interface .................................... 9  
2.3 Example of Interacting with a Computer in a 3D Space .................. 10  
2.4 Human-Computer Interface Example Approaches .......................... 10  
2.5 Examples of Stereo Systems .................................................... 11  
3.1 Gaussian Filtered Tsukuba Stereo Image Pair ............................... 12  
3.2 Example of Row Profiles Extracted from Tsukuba Image Pair .......... 13  
3.3 Example of Discontinuity ....................................................... 13  
3.4 Example Matching Process of Original Profile Shape-Matching Algorithm ....................................................... 16  
3.5 Example of Candidate Point Selection ................................. 17  
3.6 Example Matching Process of Modified Profile Shape-Matching Algorithm ....................................................... 20  
4.1 Example of a Mask ................................................................. 24  
4.2 Example of a Morphed Mask ................................................... 25  
4.3 Example of a Filtered Input Image ......................................... 25  
4.4 Detected Face Region .......................................................... 26  
4.5 Labeled Human Body Blob .................................................... 26  
4.6 Flow Chart of Human Pose Estimation .................................. 27  
4.7 Examples of Haar Features ..................................................... 28  
4.8 Cascade of Classifiers for Face Detection ............................. 29  
4.9 Human Skeleton Model ........................................................ 30  
4.10 Anthropometric Ratios of Typical Human Body .......................... 31  
4.11 Example of Shoulder Points Estimation ................................... 32  
4.12 Example of Cropping Arms .................................................. 32  
4.13 Example of Skeleton Fitting ................................................ 33  
4.14 Example of Fingertips Localization ..................................... 34  
4.15 Flow Chart of Hand Gesture Estimation ................................ 35  
4.16 Example of Finding Convex Hull ............................................ 37  
5.1 Stereo Camera System .......................................................... 39  
5.2 Two Accuracy Measurement Cases ....................................... 40  
5.3 Ground Truth and Resulting Disparity Maps of Tsukuba ................. 42  
5.4 Results of Human Pose Estimation 1 .................................... 44  
5.5 Results of Human Pose Estimation 2 .................................... 45  
5.6 Results of Human Pose Estimation 3 .................................... 46  
5.7 Results of Human Pose Estimation 4 .................................... 47  
5.8 Defined Gestures for Control ............................................... 49  
5.9 Results of Finger Gesture Estimation 1 .................................. 50  
5.10 Results of Finger Gesture Estimation 2 .................................. 51
CHAPTER 1. INTRODUCTION

Humans have interacted with computers through keyboards and mice for decades. People hold a mouse, drag, and click it, then type in the commands on a keyboard. In recent years, the new method for interacting with a computing device is touching the screen with fingers. This new method makes the communication between human and machine more intuitive. However, screen contact is limited to interactions on a 2D plane, in this case, the screen. With stereo vision, the method for controlling systems can be expanded to a 3D space. Different commands can be assigned according to the 3D information of a user’s movement and gesture.

For example, the Microsoft Kinect is a newly developed human-computer interface for entertainment (Figure 1.1). Instead of holding a gamepad, gamers can play games by moving their limbs to control the characters in the games. This novel way of playing games makes people feel they are really in the games. Another application example of a human-computer interface is one used for safety purposes. A stereo vision system installed in a car can detect and interpret the driver’s finger movements and gestures to control some functions of the car and a smart phone or tablet (Figure 1.2). The visual demand for controlling those devices can be reduced. Using this system to minimize distraction, people can drive a car more safely.

A great deal of research has been conducted in the field of stereo vision. Many new approaches are developed to solve stereo vision problems, and the existing methods are constantly improving. These efforts are focused on increasing accuracy and reducing processing time. Like other computer vision problems, the accuracy of an algorithm is affected by the lighting and object textures. A higher resolution image contains more information to generate a more accurate result but slows down the process. Furthermore, the computation complexity is also an issue that affects the accuracy and processing speed. As a result, speed and accuracy take a stand against each other. It is a challenging task to be able to improve both.
The proposed applications require a stereo vision algorithm that can be implemented in an embedded system. Most embedded systems are designed for specific tasks and usually have
limitations on resources. For example, they use batteries as a power source, have limited memory spaces, must be light weight for mobile applications, and require simple computation processes for real-time performance. As a result, almost all embedded systems are usually considered resource-limited systems. For the proposed applications, the system must produce the result in real time with adequate accuracy for analysis.

We recently developed a new stereo vision algorithm called Intensity Profile Shape Matching for 3D human gesture analysis and obstacle detection. This algorithm does not require complex computations that slow down the process. It extracts image intensity values from the same image rows of two images from a stereo image pair and matches the intensity profile shape row by row. After it iterates through the entire images, the algorithm produces a disparity map for gesture analysis and obstacle detection in real time.

Once the disparity map is calculated, a disparity range is set to remove objects that are not within the desired distance range in front of the cameras. In other words, object detection is not based on an object’s color, intensity, or texture but by its distance from the cameras. This approach reduces the effect of intensity and color differences caused by lighting variations.

This algorithm, like others, faces a trade-off between accuracy and processing speed. Higher accuracy requires more computation resources and longer processing time. For the proposed human pose estimation application, we are able to find a balance between accuracy and processing speed.

In this thesis, we modify the original Intensity Profile Shape Matching Algorithm, and then implement it for human pose estimation and finger tracking. We also introduce algorithms for building human skeleton model and fingertip localization. After the proposed stereo vision algorithm generates the 3D information, we combine the 3D information with the human skeleton model and fingertip localization to track and interpret the gesture of the controlling agents, which are the human body and fingers.

Chapter 2 discusses the qualifications and comparisons of real-time and near real-time stereo vision algorithms. It also includes the advantages and disadvantages of different existing approaches for human-computer interface. The original and modified profile shape matching algorithms are discussed in Chapter 3, which explains how to select points for the matching process, the method for finding a correspondence pattern, and post-processing for noise removal. Chap-
Chapter 4 explains the method to build the human skeleton model for pose analysis, and the method for locating the fingertips for hand gesture estimation. Chapter 5 shows and compares the results of the original and modified algorithms, which also includes the parameters for the proposed algorithm, metric for accuracy measurement, hardware and implementation. Section 5.4 presents the resulting images for the Tsukuba image stereo pair produced by both algorithms. It also shows the robustness of the proposed method by adding Gaussian noise to one of the input images. Section 5.5 presents the results of the proposed algorithm for segmenting the human body from the background and building its skeleton model. Section 5.6 shows the results of using this algorithm for hand gesture estimation. Conclusion and future work are discussed in Chapter 6.
CHAPTER 2. BACKGROUND

2.1 Processing Speed of Real-time Stereo Vision Algorithms

Stereo vision algorithms can be categorized into global and local methods. Gupta and Cho [1] defined local algorithms as statistical methods that usually depend on correlation between stereo image pairs. Global algorithms are based on explicit smoothness assumptions that are solved through various optimization techniques. They also mentioned that global algorithms are impractical for real-time systems due to their computation complexity [1], even though the most accurate stereo vision algorithms in the literature are generated by some kinds of global energy minimization algorithms. Since the proposed human pose estimation requires that the results can be generated in real time on a general-purpose CPU, this thesis is focused on the development of real-time stereo vision algorithms using a local method.

In [2] the authors defined a function

\[ Mds/s = \frac{W \times H \times D}{t} \times \frac{1}{1000000}, \]  

(2.1)

where \( W \) and \( H \) denote the width and height of an image, respectively. \( D \) represents the maximum disparity range in the image. Equation 2.1 determines the runtime performance given in millions of disparity evaluations per second \((Mde/s)\). For Tsukuba stereo image pair from the Middlebury dataset, an algorithm which achieves more than 10 \( Mde/s \) is defined as near real-time, and is defined as real-time if the algorithm has greater than 53 \( Mde/s \) on general-purpose CPUs. Images that have a larger size and disparity range similar to Cones or Teddy, the algorithms that produce a result in 1 second (1 fps) are defined as near real-time. An algorithm classified as real-time should be able to generate the result in 33 ms (30 fps). Table 2.1 lists the algorithms that have the processing speed of 10 \( Mde/s \) or higher, which produce a disparity map in 1 second by using an image pair with a similar disparity range and size to the standard Teddy and Cones stereo pairs.
Table 2.1: Comparison of disparity computations per pixel for real-time algorithms [2].

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>t (ms)</th>
<th>$W \times H$ (disp)</th>
<th>$Mds/s$</th>
<th>Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>SADL [3]</td>
<td>87.9</td>
<td>512 x 512(48)</td>
<td>143.2</td>
<td>P4 3.2GHz</td>
</tr>
<tr>
<td>RTCensus [4]</td>
<td>77.6</td>
<td>450 x 375(60)</td>
<td>130.5</td>
<td>Core 2 T7200 2.0GHz</td>
</tr>
<tr>
<td>SADRec [3]</td>
<td>97.6</td>
<td>512 x 512(48)</td>
<td>128.9</td>
<td>P4 3.2GHz</td>
</tr>
<tr>
<td>Modified ProfileShape</td>
<td>14</td>
<td>384 x 288(16)</td>
<td>126</td>
<td>Phenom II 2.8GHz</td>
</tr>
<tr>
<td>SADLR [3]</td>
<td>109.6</td>
<td>512 x 512(48)</td>
<td>114.8</td>
<td>P4 3.2GHz</td>
</tr>
<tr>
<td>ProfileShape [5]</td>
<td>16</td>
<td>384 x 288(16)</td>
<td>110.5</td>
<td>Phenom II 2.8GHz</td>
</tr>
<tr>
<td>SparseCensus [6]</td>
<td>100</td>
<td>450 x 375(60)</td>
<td>108</td>
<td>Core2 2.5GHz</td>
</tr>
<tr>
<td>RTDP [7]</td>
<td>18</td>
<td>384 x 288(16)</td>
<td>98.3</td>
<td>AMD Athlon 2800</td>
</tr>
<tr>
<td>SNCC [8]</td>
<td>140</td>
<td>450 x 375(60)</td>
<td>77.1</td>
<td>Std PC 3.0GHz</td>
</tr>
<tr>
<td>SADMW5 L [3]</td>
<td>166</td>
<td>512 x 512(48)</td>
<td>75.5</td>
<td>P4 3.2GHz</td>
</tr>
<tr>
<td>SADMW5 Rec [3]</td>
<td>169.8</td>
<td>512 x 512(48)</td>
<td>74.1</td>
<td>P4 3.2GHz</td>
</tr>
<tr>
<td>SADMW5LR [3]</td>
<td>169.8</td>
<td>512 x 512(48)</td>
<td>74.1</td>
<td>P4 3.2GHz</td>
</tr>
<tr>
<td>SADDP [3]</td>
<td>169.8</td>
<td>512 x 512(48)</td>
<td>74.1</td>
<td>P4 3.2GHz</td>
</tr>
<tr>
<td>DistinctSAD [9]</td>
<td>25</td>
<td>320 x 240(16)</td>
<td>49.1</td>
<td>PIII 800MHz</td>
</tr>
<tr>
<td>CSBP [10]</td>
<td>3550</td>
<td>800 x 600(300)</td>
<td>40.5</td>
<td>Core2 2.5GHz</td>
</tr>
<tr>
<td>RealtimeVar [11]</td>
<td>285</td>
<td>450 x 375(60)</td>
<td>37.8</td>
<td>P4 2.83GHz</td>
</tr>
<tr>
<td>OpenCV SGBM [3]</td>
<td>459.4</td>
<td>512 x 512(48)</td>
<td>27.4</td>
<td>P4 3.2GHz</td>
</tr>
<tr>
<td>SAD [12]</td>
<td>505</td>
<td>450 x 375(60)</td>
<td>21.3</td>
<td>n/a</td>
</tr>
<tr>
<td>Census [12]</td>
<td>582</td>
<td>450 x 375(60)</td>
<td>18.5</td>
<td>n/a</td>
</tr>
<tr>
<td>FastAggreg [13]</td>
<td>600</td>
<td>450 x 375(60)</td>
<td>18</td>
<td>Core Duo 2.14GHz</td>
</tr>
<tr>
<td>OpenCV SAD [14]</td>
<td>673</td>
<td>450 x 375(60)</td>
<td>16</td>
<td>P4 3.0GHz</td>
</tr>
<tr>
<td>OptimizedDP [15]</td>
<td>800</td>
<td>450 x 375(60)</td>
<td>13.5</td>
<td>Std PC 1.8GH</td>
</tr>
<tr>
<td>BilateralSAD [16]</td>
<td>100</td>
<td>320 x 240(16)</td>
<td>12.2</td>
<td>P4 2GHz</td>
</tr>
<tr>
<td>SegTreeDP [17]</td>
<td>863</td>
<td>450 x 375(60)</td>
<td>11.7</td>
<td>P4 2.4GHz</td>
</tr>
<tr>
<td>SAD-MW [18]</td>
<td>213</td>
<td>320 x 240(32)</td>
<td>11.5</td>
<td>PII 450MHz</td>
</tr>
<tr>
<td>DSI atrix/SAD [19]</td>
<td>161</td>
<td>384 x 288(16)</td>
<td>10.9</td>
<td>P4 2.66GHz</td>
</tr>
<tr>
<td>HistoAggr [20]</td>
<td>980</td>
<td>450 x 375(60)</td>
<td>10.3</td>
<td>n/a</td>
</tr>
<tr>
<td>MeanSAD [21]</td>
<td>218</td>
<td>384 x 288(20)</td>
<td>10.1</td>
<td>PIII 800MHz</td>
</tr>
<tr>
<td>SemiGlob [22]</td>
<td>1000</td>
<td>450 x 375(60)</td>
<td>10.1</td>
<td>Xeon 2.8GHz</td>
</tr>
<tr>
<td>RSR/TSDP [23]</td>
<td>320</td>
<td>256 x 256(30)</td>
<td>6.1</td>
<td>PIII 500MHz</td>
</tr>
</tbody>
</table>
2.2 Real-time Stereo Algorithm for Resource Limited Systems

Most research work in stereo vision focuses on improving and optimizing the stereo vision algorithms to produce results in real time. Many of these attempts rely on hardware that is able to run processes in parallel such as multiple cores and SIMD instructions to utilize the parallelizable nature of stereo vision algorithms to achieve results in real time. Some published studies developed algorithms for embedded systems, e.g., field-programmable gate arrays (FPGAs), digital signal processing, application-specific integrated circuits, and low power multicore processors. These embedded systems may have resource limitations, such as power consumption, memory usage, payload capacity, and processing speed [24–29].

A method for comparing the processing speed of resource limited applications using different types of hardware was proposed in [2]. Equation 2.2 is defined for computing the ratios of the normalized runtimes to the average of the normalized runtimes.

\[
\text{ratio}_i = \frac{K_i}{\text{avg}} , \quad K_i = \frac{\text{Mdes}/s}{\text{PassMark}_i} , \quad \text{avg} = \frac{\sum_{i=0}^{N} K_i}{N} .
\]  

(2.2)

PassMark is the benchmark value listed in [30]. The benchmark value for each CPU was gathered from PassMark which is collected from the users and the internal tests.

Figure 2.1 shows runtimes of the real-time stereo algorithms with sufficient hardware information normalized by the benchmark value of AMD Phenom II X6 1055T. The proposed algorithm which is the ModifiedProfileShape has the best runtime without SIMD optimization. In addition, ModifiedProfileShape has better runtime than its original version named ProfileShape.
Figure 2.1: Normalized runtimes of all real-time stereo algorithms that provide sufficient hardware detail [2]. (* indicates the algorithm is optimized by using SIMD instructions.)
2.3 Human-Computer Interface

The purpose of human-computer interface is to define convenient and efficient ways to control computer functions. For example, decades ago, people needed to type in a long complicated command for moving a file to a different folder. Not too long after that, all it is needed is to simply click on a mouse and drag the file icon into the target folder. Nowadays, a mouse is not needed to give a computer the same command. All it needs is touching and dragging the icon on the screen with our fingers. Figure 2.2 illustrates the evolution of human-computer interface.

Figure 2.2: The evolution of the human-computer interface.

Human-computer interface research focuses on making communication between humans and computers more intuitive. Recently, a great deal of research and development has gone into expanding the method of communicating with computer systems from a 2D plane to a 3D space. People no longer need to put a finger on a screen, but instead move their finger freely in space. Figure 2.3 is a scene from the first Iron man movie. The character interacts with the computer system without holding a mouse or touching a screen.
Almost all techniques for this purpose utilize a similar idea: detect and interpret a human body’s movement. One approach is to attach sensors to the body, and analyze the signals sent back from those sensors. Some systems require the user to wear colored markers on their body [31], and let the system track the markers. The example of these are shown in Figure 2.4. However, it is impractical to always ask the controller to attach sensors or wear special clothing with markers.

A proposed infrastructure-less interface is a stereo vision system. The examples of stereo systems are shown in Figure 2.5. By choosing the desired disparity range we can segment a body in a range. The stereo vision algorithms based on color space [32, 33] might be interfered with different lighting conditions because objects will all appear white under a strong light no matter their original color.

A popular 3D system, the Microsoft Kinect, developed for the video game industry has been used in the stereo research field. It has been used to achieve non-contact gesture interface [34].
Although it provides adequate accuracy, it is easily affected by sunlight because of its infrared Time of Flight (ToF) mechanism. As a result, it is not suitable for the environment that is often affected by sunlight.

Figure 2.5: Examples of stereo systems.

The problems above can be alleviated by using the stereo vision system with two cameras. By choosing a specific algorithm, the system can tolerate lighting differences. In addition, the stereo vision algorithm must return results for real-time analysis. The proposed algorithm, which is the modified profile shape-matching algorithm, can achieve these.
CHAPTER 3. PROFILE SHAPE MATCHING ALGORITHM

When observing the row intensity profile of stereo image pairs we are inclined to match the identical patterns and shapes. Figure 3.2 shows the Gaussian-filtered row intensities from the Tsukuba stereo image pair in Figure 3.1(a) and Figure 3.1(b). It is obvious that the profiles of the two rows share a similar pattern. Upon further examination, one profile actually contains spans not found in the other. Examples of these subtle differences that represent discontinuities in the disparity map are highlighted in Figure 3.3. The algorithm assumes that all points belong to an intensity profile shape, which are bounded by discontinuities, and all points in corresponding shapes in the two images are on the same depth plane. Similar assumptions are found in [35–38]. It also assumes a point can only match with one unique corresponding point in the other image, which is known as the uniqueness constraint.

![Gaussian filtered Tsukuba stereo image pair](image1)

(a) (b)

Figure 3.1: Gaussian filtered Tsukuba stereo image pair (a) is the left image and (b) is the right image.
Figure 3.2: Row profiles extracted along the white line in Figure 3.1(a) and Figure 3.1(b).

Figure 3.3: Red rectangle highlights the discontinuity between two profiles.
3.1 Original Profile Shape Matching Algorithm

The original profile shape matching algorithm has the following steps:

1. Convert input image pair into grayscale and smooth them using a Gaussian filter.

2. Select the local maximum points as vertices from each row in a stereo image pair.

3. Based on the points selected in Step 2, identify and match similar shapes in the same row. At the end of this step, every pixel is assigned a disparity value.

4. Apply vertical smoothing as post processing to remove streaks that are common in line-wise matching.

The Gaussian filter in Step 1 smoothes and reduces noise contained in images. This noise-smoothing step increases the matching accuracy. In Step 2, the point that has higher pixel value than its two neighbors on the same row is selected as a vertex. The ordered set $V^k \ni v_m^k$ is the selected vertices in each row where $m$ represents the index of a vertex, and $k$ denotes the left (L) or right (R). Equation 3.1 denotes the computation of the gradient $\nabla^k$ between two adjacent pixels.

$$\nabla^k [x] = I^k[x+1] - I^k[x].$$  (3.1)

Step 3 starts from the vertex which has the highest intensity value in the left image compared against all selected vertices in the right image within the maximum desired disparity, $d_{max}$. The vertices in $V^L$ and $V^R$ are compared for matching by expanding a span around each vertex. The matching region stops expanding when the difference of gradient is greater than the initial threshold. Larger thresholds are set for further iterations. Vertices sharing the longest span are considered to be in the same shape. The notation, $S_n(v^L_n, p_i^L, q_i^L, v^R_n, p_i^R, q_i^R)$, where $n$ is the shape label, $p_i^L$ and $q_i^L$ are the starting and ending pixel index for left (L) or right (R) image. The disparity values are the difference of $v^L_n$ and $v^R_n$. Equation 3.2 shows that the gradient difference between $p_i^k$ and $q_i^k$ should be less than the threshold $t_i$.

$$S_n(v^L_n, p_i^L, q_i^L, v^R_n, p_i^R, q_i^R) = [p_i^k, q_i^k] s.t$$

$$\forall x : p_i \leq x \leq q_i, |\nabla^R[x] - \nabla^L[x+d_n]| < t_i.$$  (3.2)
Because the threshold is lower at the beginning, most of the pixels tested are not assigned to any shape. The threshold is increased during successive processes, and all the pixels tested will end up be assigned a shape label. This process of profile matching is summarized in Algorithm 1.

Figure 3.4 shows an example of the matching process. Red circles in Figure 3.4(a) indicate selected vertices. The algorithm compares the vertex in the left profile against three vertices in the right profile, and the threshold for comparison increases after each iteration until it reaches a preset value. Spans shared by paired vertices are highlighted in Figure 3.4(b), 3.4(c), and 3.4(d). The vertex pair has the longest span is considered as the best match. This is shown in Figure 3.4(c).

```plaintext
for row in image do
    for vertex, v_L^n in V_L do
        for vertex, v_R^m in V_R where v_L^n − v_R^m ≤ d_max do
            while |∇^R[q^k]_0 − ∇^L[q^k]_0| < t_0 do
                Increment q^k
            end
            while |∇^R[p^k]_0 − ∇^L[p^k]_0| < t_0 do
                Decrement p^k
            end
        end
        Set S_n(v_L^n, p_L^0, q_L^0, v_R^n, p_R^0, q_R^0)s.t. arg max_s(q_R^0 − p_R^0)
    end
for t_i in thresholds do
    for vertex, v_R^n in V_R do
        while |∇^R[q^k_i] − ∇^L[q^k_i]| < t_i do
            Increment q^k
        end
        while |∇^R[p^k_i] − ∇^L[p^k_i]| < t_i do
            Decrement p^k
        end
        if every pixel is assigned to a shape then
            Continue to next row of image
        end
    end
end
```

**Algorithm 1: Profile Matching**
Figure 3.4: The circled points in (a) are selected vertices. The paired vertices in (c) share the span that is longer than the spans in (b) and (d). As a result, the paired vertices in (c) are the best match.

3.2 Modified Profile Shape Matching Algorithm

The new algorithm is based on the original profile shape matching algorithm [39]. It includes five steps:

1. Convert input image pair into grayscale and smooth them using a Gaussian filter.

2. Select candidate points in the left image.

3. Find the corresponding points in the right image.

4. Assign a shape label to every point in the image.

5. Apply vertical smoothing as post process for the entire resulting disparity map.

Like the original algorithm, we convert input images into gray scale and apply a Gaussian filter to perform pre-processing in Step 1 to smooth out noise. After the images are smoothed
by the Gaussian filter, the algorithm needs to locate points to match. Instead of picking points randomly, this new version of algorithm selects points to match based on the intensity changes between two neighboring pixels. Here, we call the selected point a candidate point. This second step of the algorithm uses two thresholds. If the absolute intensity difference between two adjacent points is less than the Minimum Step Threshold then it will be ignored. Points will only be selected when the accumulation is greater than the Accumulated Change Threshold. After that, the process will start anew at the next point on the row. The advantage of this method is that the number of candidate points selected from flat surfaces can be reduced, and instead more candidate points will be selected from surfaces with abundant variances.

Figure 3.5 shows an example of the candidate point selection process. In this example, we assume the Minimum Step Threshold is equal to 2 and the Accumulated Change Threshold is equal to 5. The changes highlighted by red lines are ignored because they are smaller than the Minimum Step Threshold. Two circled points are selected because the accumulation is greater than the Accumulated Change Threshold at these points.

![Figure 3.5: Example of the candidate selection process with Minimum Step Threshold of 2 and Accumulated Change Threshold of 5.](image)

In Step 3, for each candidate point in the left image, its corresponding point in the right image must be identified. We introduce a voting mechanism to achieve this. Every candidate point
in the left image is treated as a starting point of the search and expanded a span of a certain length, \( l \). The slopes \( S_a^x \) of two consecutive nodes is calculated as shown in Equation 3.3.

\[
\frac{v_a^n - v_a^{(x-1)}}{l} = S_a^x, \tag{3.3}
\]

where \( v_a^n \) is the intensity value of the \( x^{th} \) vertex, and \( a \) is either L (the left image) or R (the right image).

We define a disparity range with a lower bound \( LB \) and an upper bound \( UB \). We repeat the same process for every point in the right image that lies within the desired range. Compare the sign of \( s_L^x \) and \( s_R^x \); if they have the same sign and the absolute difference is less than the Same-Sign Threshold \( f \), then \( s_L^x \) and \( s_R^x \) are considered having an identical pattern and vote count is incremented. When \( s_L^x \) and \( s_R^x \) have different signs, we use a threshold called the Zero-Crossing Threshold and denoted as \( f_{zerocrossing} \), which is a positive number smaller than \( f \). The starting point of a span that has the highest number of votes is chosen as the corresponding point in the right image. This process is summarized in Algorithm 2.

\begin{verbatim}
for i = 1 to number of picked points in the left image do
    current max vote = 0
    for j = \( P_i^l - LB \) to \( j = P_i^l - UB \) do
        vote = 0
        for k = \( P_j^R - l \) to \( j = P_j^R + l \) do
            if \( s_L^x, s_R^x \) have the same sign and \( |s_L^x - s_R^x| < f \) then
                Increment vote
            end
            if \( s_L^x, s_R^x \) have different sign and \( |s_L^x - s_R^x| < f_{zerocrossing} \) then
                Increment vote
            end
        end
        if vote > current max vote then
            current max vote = vote
            corresponding, point in the right image = \( P_j^R \)
        end
    end
end
\end{verbatim}

\textbf{Algorithm 2: Finding Corresponding Points in the Right Image for Each Row}
The voting mechanism matches patterns based on the gradient of shape, not the intensity value. It also ensures the slopes have the same orientation by testing the signs of the slopes. This unique voting mechanism is robust even if the input image pair is affected by a small noise.

An example of finding corresponding points is shown in Figure 3.6. We use the same profile in Figure 3.5 as the left profile. In this example, we assume the Zero-Crossing Threshold is equal to 1 and the Same-Sign Threshold is equal to 2. The algorithm starts by comparing the two circled points in Figure 3.6(a). It computes and compares the slopes near these two points. The highlighted slopes in the left profile are 2 and 3. The first highlighted slope in the right profile is 3, and the difference between the two first slopes from the left and right profiles is less than the Same-Sign Threshold. As a result, it receives 1 vote. However, the two second slopes from the left and right profiles are not matched since the difference is greater than the Zero-Crossing Threshold. This process continues along the right profile. No match is found until to the second-to-last point as shown in Figure 3.6(b). The differences of left and right slopes are both smaller than the Same-Sign Threshold, and the vote is the highest in the right profile. Therefore, they are considered a good match.

In Step 4, the pairing points, $V^L_m$ and $V^R_m$, where $m$ represents the index of point pairs, are used as the roots and grow forward and backward in each row. $[q^a_m, p^a_m]$ is used to represent a span, where $q^a_m$ denotes the beginning index of a span, and $p^a_m$ denotes the end of a span. Equation 3.4 and Equation 3.5 represent the slope calculation between $V^a_m$ and its neighbor.

$$V^a_{m-q} - v^a_{m-q-1} = S^a_q.$$  \hspace{1cm} (3.4)

$$v^a_{m+p} - v^a_{m+p+1} = S^a_p.$$ \hspace{1cm} (3.5)

In addition, we define two sets of threshold $T$ and $T_{\text{zerocrossing}}$ for the matching process. The subsets of them are $t_n \in T_{\text{zerocrossing}}$ and $t_n^{\text{zerocrossing}} \in T_{\text{zerocrossing}}$. As in the previous step, when one of the slopes of $S^L_q$ and $S^R_q$ is zero, or if they have the same sign and their difference is less than $t_n$, then the same shape label is assigned to these two neighboring points. If $S^L_p$ and $S^R_q$ have different signs, we use a different threshold $t_n^{\text{zerocrossing}}$ to evaluate the difference instead. Again, $t_n^{\text{zerocrossing}}$ is usually smaller than $t_n$. The spans $[q^a_m, p^a_m]$ in both images grow longer because both
Figure 3.6: An example of finding the corresponding point in the right profile the Zero-Crossing Threshold of 1 and the Same-Sign Threshold of 2.

of these thresholds increase in each iteration. This process assigns a shape to only the points that have not yet been assigned to any shape. This process is summarized in Algorithm 3.
for $t_n$ in threshold set $T$ do
    error = 0
    for $m = 1$ to number of picked point pairs do
        while error == 0 do
            for $i = 1$ to number of picked points in both images do
                if ($S^L_q, S^R_q$ have the same sign and $|S^L_q - S^R_q| < t_n$) or ($S^L_q, S^R_q$ have different sign and $|S^L_q - S^R_q| < t_n^{zerocrossing}$) then
                    Increment $q$
                else
                    error = 1
                end
            end
        end
        error = 0
    while error == 0 do
        for $i = 1$ to number of picked points in both images do
            if ($S^L_p, S^R_p$ have the same sign and $|S^L_p - S^R_p| < t_n$) or ($S^L_p, S^R_p$ have different sign and $|S^L_p - S^R_p| < t_n^{zerocrossing}$) then
                Decrement $p$
            else
                error = 1
            end
        end
    end
end

Algorithm 3: Matching Patterns for Each Row

The final step of our algorithm is a vertical smoothing process, which is the same as the original profile shape matching algorithm. Since the matching process is a row operation, the information contained in the image columns is not considered in the previous steps of the process. The vertical smoothing process takes a majority vote of the disparities of the target pixel and five pixels above and five pixels below it and assigns the majority disparity to the target pixel.

Due to the noise in the images, the vertices selected by the original profile shape matching algorithm in the left image may not find the correct corresponding vertex in the right image. Instead of picking the vertices in both images and then finding the correspondences within the disparity range, this modified algorithm selects vertices in the left image and does the matching process
exhaustively on every point within the disparity range. Although the accuracy is higher than the original version, the modified algorithm requires a slightly longer processing time.

In the process of the original algorithm, each vertex needs 6 subtractions for comparing gradients. In addition, no more than half of the points can be vertices within a disparity range, $d_{\text{max}}$. The number of the vertices in one row cannot be more than half of the row. As a result, in the worst case, the number of operations per pixel is approximately $1/2 \times 6 \times 1/2 \times d_{\text{max}}$ subtractions plus $d_{\text{max}}$ comparisons.

The modified algorithm selects a vertex point when the accumulated change is greater than the accumulated change threshold. Every selected point expands a span with $l$ pixel width, compared against pixels within $d_{\text{max}}$ which are also expanding the spans with the width $l$. A total of $3 \times (l - 1)$ subtractions and $2 \times l$ comparisons are needed for comparing slopes in two spans. $d_{\text{max}}$ comparisons are needed for determining the best match. In the worst case scenario, we assume all differences between pixels are greater than the accumulated change threshold. In other words, we assume every point is selected as a vertex point. The number of operations per pixel is $3 \times (l - 1) \times d_{\text{max}}$ plus $2 \times l + d_{\text{max}}$ comparisons.

The proposed algorithm requires more operations per pixel than the original version. However, it does not need to sort the selected vertices and start the matching process with the vertex that has the highest intensity value in a row. Additionally, the proposed algorithm needs fewer thresholds than the original version. Depending on the amount of selected points, it may need fewer subtractions and comparisons in total on some images. In other words, it can achieve higher accuracy and close or even faster running speeds. The result of the proposed algorithm is discussed in Chapter 5.
CHAPTER 4.  HUMAN POSE AND HAND GESTURE ESTIMATION

This chapter demonstrates methods that utilize the disparity results produced by the proposed algorithm. Human pose and hand gesture estimation are chosen to prove that the proposed algorithm is suitable for practical use. For human pose estimation, we build the human skeleton model to fit the limbs. The 3D relations between the joints in the model are used to estimate the pose. For hand gesture estimation, the fingertips need to be located first, and then the gesture estimation can be done by interpreting relation between fingertips in 3D space. The hand gestures are then interpreted and sent as the inputs to control a smart phone or tablet computer.

4.1 Human Pose Estimation

The disparity map generated by the proposed algorithm is used as a mask for filtering. We can then retain the object at the desired distance and remove other objects which are not in the range. The object in the proposed application is the human body. In [40], the authors used a depth camera which uses infrared to measure the distance. Although a stereo system using the depth camera has higher accuracy and faster runtime compared to most stereo systems with two cameras, it cannot be used in the environment that is affected by sunlight. The performance of the proposed stereo vision algorithm is not affected by sunlight. It is able to calculate the disparity map as long as the intensity profile is not uniform or the profile shapes are discriminable.

The segmented human body produced by the proposed algorithm has broken parts. There are also mismatched parts in the resulting images. An example of this is shown in Figure 4.1. These noise sources affect the accuracy of the pose estimation.

Since the object is a human, we include a face detection algorithm [41] to assist locating the blob of the body. The holes caused by noise may appear on the human head in the resulting image which affects face detection accuracy. As a result, after creating the mask, we apply a morphological operation to the mask to fill out the holes, and then filter the input image by the
mask. In Figure 4.2, the holes appearing in the face part in Figure 4.1 are filled. The binarized image in Figure 4.2 can then be used as a mask to filter out the background and to detect the complete human face in the filtered image as shown in Figure 4.3.

Figure 4.4 is the face detected by Haar face detection [41] in the filtered image. Once we obtain the coordinate of the face, the flood fill algorithm is applied to find the connected components of the face. The gray part in Figure 4.5 is the labeled blob of the human body. Using this human body blob, we are able to build a human body model for pose estimation. Haar face detection also returns the size of the detected face. This information can be used to build the skeleton model to fit the shoulder and two arms. Disparity values will then be assigned to the corresponding joints of the skeleton model. The depths and the relative position of the joints are used for estimating human pose. The flow chart of the above processes is shown in Figure 4.6.
Figure 4.2: The mask created by applying morphology operation on the threshold and binarized disparity map.

Figure 4.3: The input image filtered by the mask.
Figure 4.4: The face image detected in the filtered input image.

Figure 4.5: The result of the flood fill algorithm using the face coordinate as the seed, where the gray part indicates the human body blob.
Figure 4.6: The flow chart of human pose estimation.

- Obtain the target disparity map
- Create a mask by thresholding and binarizing the target disparity map
- Use the morphological operation to close holes in the mask
- Use the mask to filter the input image
- Detect the human face in the filtered image
- Set the coordinate of the face as a seed and apply flood fill algorithm
- Estimate the joints on upper human body and build a human skeleton model
- Assign each joint point a disparity value according to the disparity map
4.1.1 Haar Face Detection

Haar feature detection is done by using Haar features as shown in Figure 4.7 to determine if a region of an image contains the feature of interest. To determine the presence of a Haar feature, the detector subtracts the average pixel value in the black region from the average pixel value in the white region and compares the difference to a threshold. A Haar feature is detected when the difference is greater than the threshold. A human face can be represented by the combination of Haar features.

The authors in [41] cascade a series of classifiers as a filter chain as shown in Figure 4.8. Each classifier determines whether the region of interest contains a Haar feature forming a face. The region that passes through all the classifiers is said to be a face.
Figure 4.8: Each red node is a classifier of the face detector.
4.1.2 Human Body Model

The human skeleton model can be represented by 6 vectors as shown in Figure 4.9. The ratios between human parts are fixed and determined based on the NASA Anthropometric Source book [42] (Figure 4.10).

![Human Body Model Diagram]

Figure 4.9: The human body model used for skeleton fitting.

Depending on the size and the location of the detected face, the green points in Figure 4.9 including one head point, and two shoulder points can be estimated as shown in Figure 4.11. Based on the shoulder points, the left and the right arms are cropped based. It is shown as two yellow regions in Figure 4.12. And then the linear regression algorithm is applied to find the linear function of the arms, which are represented by four red points in Figure 4.9. An example of human skeleton fitting is shown in Figure 4.13.

After all points in the upper body have been located, the disparity values of these points are assigned to them for gesture estimation. In addition, we can then define specific gestures of the body as inputs to an electronic device for control purposes.
Figure 4.10: Anthropometric ratios of typical human body.
Figure 4.11: Two shoulder points can be estimated by the size and the position of the detected face.

Figure 4.12: Two arms are cropped based on the shoulder points.
4.2 Hand Gesture Estimation

Unlike human pose estimation, we cannot locate a point as a “seed” for a flood fill algorithm to find the target blob. For hand gesture estimation, the background subtraction is applied for extracting the hand from the background, and then the convex hull of the hand can be determined by using the method in [43]. Since the vertices of the convex hull may not always be formed by fingertips, we select a region of interest and only detect the vertices in this region. An example is shown in Figure 4.14. All five fingertips are shown in red circles. According to the resulting disparity map, disparity values will be assigned to these fingertip locations. Figure 4.15 shows the flow chart of this process.

The depth and the relative position of the fingertips in the detecting region are utilized to estimate hand gesture. Each estimated gesture is treated as a specific function to control a smart phone or tablet computer.
Figure 4.14: (a) shows the hand that isolated from the background. The white line in (b) is the convex hull of the hand. The red circles in (c) are the fingertips forming the convex hull.
Figure 4.15: The flow chart for hand gesture estimation.
4.2.1 Convex Hull

The convex hull of a point set is the smallest convex set that contains all points in the point set. The method for finding the convex hull of a hand is called the Graham scan in [43]. The algorithm first finds the starting point, the point with the highest y-coordinate, and finds the point that has the lowest x-coordinate if more than one point has the highest y-coordinate. In the second step, it sorts the points in increasing order of the angles between the line connecting the point and the starting point and the x-axis. In the last step, the algorithm processes the points in order. Depending on the direction of the turn it takes from the two previous points to the current point. The second-to-last point is not on the convex hull and should be removed if it makes a right turn, and then connects the third-to-last point with the last point. It backtracks and checks the direction of the turn again until it makes a left turn which means the second-to-last point is on the convex hull. The algorithm stops when it forms a closed shape.

An example is shown in Figure 4.16. The points in Figure 4.16(a) are sorted in alphabetical order. Point A is the starting point. The algorithm connects points B and C in Figure 4.16(b) and 4.16(c). It makes a right turn moving from point C to point D which is shown in the red line connecting points C and D in Figure 4.16(d). As a result, point C is not part of the convex hull and is removed from the boundary in Figure 4.16(e). The algorithm only makes left turns to form a closed shape in Figure 4.16(f).
Figure 4.16: Five points are ordered alphabetically in (a). The process marches forward until it makes a right turn when connecting C and D in (d). In (e), C is removed from the convex hull. Connected points in (f) form the convex hull.
CHAPTER 5. RESULTS

5.1 Proposed Algorithm Parameters

A brute force method was applied to find the combination of parameters for Tsukuba and proposed applications. In order to improve the performance of the original algorithm, a combination of parameters for the proposed stereo algorithm that can generate the disparity map with higher accuracy and faster processing speed on Tsukuba stereo image pair was selected. For human pose and hand gesture estimation, we used the parameter combinations that generate the result in real time with sufficient accuracy. Depending on applications and 3D scenes, different combinations of parameters may have to be determined.

Kernel sizes from $5 \times 5$ to $25 \times 25$ with standard deviation values from 1 to 10 in steps of 0.5 for Gaussian filter were tested. There are also 7 parameters from Step 2 to Step 4 of the proposed algorithm which are the Minimum Step Threshold, the Accumulated Change Threshold, $f$, $f_{\text{zerocrossing}}$, $t_n$, and $t_n^{\text{zerocrossing}}$. The same brute force was used to determine the best combination of these 7 parameters. The combination of parameters that generates the best result on Tsukuba and human pose estimation was a $9 \times 9$ kernel size with a standard deviation value of 3, and the rest of parameters are listed in Table 5.1.

<table>
<thead>
<tr>
<th>Parameters</th>
<th>Tsukuba</th>
<th>Human pose estimation</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimum step threshold</td>
<td>5</td>
<td>0.5</td>
</tr>
<tr>
<td>accumulated change threshold</td>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>$l$</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>$f$</td>
<td>1</td>
<td>5</td>
</tr>
<tr>
<td>$f_{\text{zerocrossing}}$</td>
<td>0.4</td>
<td>0</td>
</tr>
<tr>
<td>$t_n$</td>
<td>1,5,8</td>
<td>0.4,0.8,1.5,2</td>
</tr>
<tr>
<td>$t_n^{\text{zerocrossing}}$</td>
<td>0.4</td>
<td>0.3,0.4</td>
</tr>
</tbody>
</table>
5.2 Hardware Details and Implementation

The original and modified profile shape matching algorithms were both implemented in C++ without any manual code optimization and executed on a machine with an AMD Phenom II 2.8GHz processor. The stereo camera system for human gesture estimation is shown in Figure 5.1. This system is built by two Pointgrey’s Flea®2 CCD cameras. The two cameras are adjusted manually so that their optical axes are close to parallel. The images are used in our algorithm without rectification to prove the algorithm’s robustness. The resolution of both input images is $640 \times 480$. The Gaussian filter, morphological operation, Haar face detection, flood fill algorithm, linear regression, and convex hull algorithm were implemented using the Open CV library. The analyzed results are interpreted as specific keys on the keyboard. An android application, ShareKM\(^1\), which maps the keyboard inputs to the software functions, was used to control a hand-held device.

Figure 5.1: The stereo camera system for human pose estimation.

\(^1\)Available at https://sites.google.com/site/droidskm/
5.3 Accuracy Measurement

The metric used for measuring the accuracy is called the bad matching pixel measurement in [44]. It computes the disparity difference of the resulting disparity map and the ground truth at the same pixel index. A bad matching pixel is a pixel whose disparity error is greater than a threshold, $\delta_d$. The percentage of bad matching pixels is calculated as

$$B = \frac{1}{N} \sum_{(x,y)} \left( |d_c(x,y) - d_T(x,y)| > \delta_d \right),$$

where $N$ is the total number of pixels, $d_c(x,y)$ is the disparity value of a pixel in the resulting image, and $d_T(x,y)$ is the disparity value of a pixel in the ground truth image.

The accuracy measurement computes the percentage of bad matching pixels in two cases: *all* and *nonocc*. In the first case, the disparity difference is calculated in the whole image except the black boundary as shown in Figure 5.2(a). The second case computes the difference of the nonoccluded region, which is shown as white areas in Figure 5.2(b). The occluded areas highlighted in black are excluded. The ground truth for both cases are obtained from the Middlebury library online.

![Figure 5.2: (a) is the area for computing the bad matching pixel in the whole image case. (b) is the area for computing the bad matching pixel for nonoccluded regions.](image-url)
5.4 Resulting Images of Tsukuba Stereo Image Pair

Tsukuba stereo image pair is a well known computer generated image pair for testing the accuracy of stereo vision algorithms. Figure 5.3(a) shows the ground truth of its disparity map. Different intensity values represent different distances from the object to the stereo camera system. Figure 5.3(b) is the result produced by the original profile shape matching algorithm. Figure 5.3(c) is the resulting image of the modified profile shape matching algorithm. Figure 5.3(d) demonstrates the robustness of the modified algorithm. In this case, a zero mean Gaussian noise with standard deviation of 2 was added to the right image. The result still demonstrates sufficient accuracy for the proposed applications.

The percentages of bad matching pixels of the original and modified profile shape matching algorithms are shown in Table 5.2. The accuracy measurement was calculated using Equation 5.1. The columns with label all show the disparity differences between the ground truth and the resulting disparity map at every pixel except the black region on the border in Figure 5.2(a). In the case of nonocc, the accuracy measurement calculates the disparity differences pixel by pixel only in the white area in Figure 5.2(b).

The labels noncc < 1 and all < 1 indicate that a pixel is counted as a bad matching pixel when its disparity difference is greater than 1. The labels noncc ≤ 2 and all ≤ 2 show that a pixel is considered as a bad matching pixel when its disparity difference is greater than or equal to 2. In all four cases, the modified algorithm has a lower percentage of bad matching pixels than the original version. The modification removes approximately 30% of bad matching pixels.

The average processing time for the entire profile shape matching algorithm, from Gaussian smoothing to vertical smoothing is 16 ms (62 fps) for the original version. The average processing time for the modified algorithm is 14 ms (66 fps) which also includes Gaussian smoothing and vertical smoothing. The modified algorithm is approximately $1.14 \times$ faster than the original version.
Table 5.2: Percentage of bad matching pixels in the resulting disparity map.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Image</th>
<th>noncc ≥ 1</th>
<th>all ≥ 1</th>
<th>nonocc &gt; 2</th>
<th>all &gt; 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original Profile shape</td>
<td><em>Tsukuba</em></td>
<td>9.6</td>
<td>11.5</td>
<td>3.2</td>
<td>5.0</td>
</tr>
<tr>
<td>Modified Profile shape</td>
<td><em>Tsukuba</em></td>
<td>6.5</td>
<td>8.1</td>
<td>2.2</td>
<td>3.6</td>
</tr>
</tbody>
</table>

Figure 5.3: (a) is the ground truth of the Tsukuba stereo image pair. (b) is the disparity map of the original profile shape matching algorithm. (c) is the disparity map of the modified algorithm. (d) is the disparity map of the modified profile shape matching algorithm with a zero mean Gaussian noise with standard deviation of 2 added to the right image of the image pair.
5.5 Resulting Images of Human Pose Estimation

This section shows that the proposed algorithm can successfully segment out the human body and build a human skeleton model in the desired disparity range. The disparity range is selected to be between 35 to 45 pixels. The background was removed because it was outside of the desired distance range. Colors were used to represent the object distances. Green is for the farthest object, then yellow, and red is the closest. The processes in Section 4.1 were implemented for human gesture estimation. In order to speed up the process, we specified a certain region for face detection. As a result, the processes only work when a face shows up in the region.

Figures 5.4(a) and 5.4(b) are the left and right input images from the stereo camera system. The human model put both hands on knees which are shown in red color in Figure 5.4(c). The red color indicates they are closer to the camera than the yellow upper body part. The color dots on the human skeleton model in Figure 5.4(d) represent the depths of each part. Figures 5.5(a) and 5.5(b) show the right arm raised and moved forward. The right palm and right elbow were at the same depth and were assigned the same red color. The right arm moved backward in Figure 5.6(a) and 5.6(b). The movement was detected and shown in Figure 5.6(c) and the resulting skeleton model is shown in Figure 5.6(d).

More results are shown in Figure 5.7. The algorithm detects the face in the blue rectangle. The color of joints in Figure 5.7(a), 5.7(b) and 5.7(c) indicates they have the same distance to the cameras. The red dot in Figure 5.7(d) at the right hand position shows the hand is closer to the cameras than other joints. Similar results can be seen in Figure 5.7(e), which shows the left arm moved forward. In Figure 5.7(f), the green dots on the right arm indicate that they are farther from the cameras than the head and the left arm is closer than the head. By analyzing the color dots in Figure 5.7, the human gesture can be determined and used for interfacing with a computing device.
Figure 5.4: (a) and (b) are the left and right input images. (c) is the resulting image after applying morphology operation. (d) is the human skeleton model for gesture estimation.
Figure 5.5: The image with the right hand raised and moved forward. (a) and (b) are the left and right input images. (c) is the resulting image after applying morphology operation. (d) is the human skeleton model for gesture estimation.
Figure 5.6: The image with the right hand raised and moved backward. (a) and (b) are the left and right input images. (c) is the resulting image after applying morphology operation. (d) is the human skeleton model for gesture estimation.
Figure 5.7: The skeleton fitting results of different gestures. All joints in (a), (b), and (c) were at the same depth plane. The right hand in (d) moved forward. The left arm moved forward in (e). In (f), the left arm moved forward with the right arm moved backward.
5.6 Resulting Images of Hand Gesture Estimation

The processes for locating fingertips in Section 4.2 were implemented for hand gesture estimation. Figures 5.9(a) and 5.9(b) are the left and right input images from the stereo camera system. In Figures 5.9, the index and middle finger had the same disparity value as the palm and arm. This is reflected in Figure 5.9(c). The gesture estimation was obtained by combining the disparity values with the locations of the fingertips. Figure 5.9(d) shows the estimation result. The tips of the index and middle finger had the same distance and were assigned the same color. The middle finger was moved forward in Figures 5.10(a) and 5.10(b). As a result, the middle fingertip was assigned yellow as shown in Figure 5.10(c). The depths of fingertips are shown in different colors in Figure 5.10(d).

Depending on the distance and the depth of the fingertips, different hand gestures can be recognized. Each defined gesture can be assigned a key input of a keyboard. Through the Android application mentioned in Section 5.2, the keys are mapped to the defined functions to control a smart phone.

For our experiments, the region of interest was an 200 × 100 area in the middle of the image, and the color representations for distances were the same as mentioned in Section 5.5. We assigned four gestures as shown in Figure 5.8 to control a smart phone: turn on/off screen, play/stop music, play previous song, and play next song. Commands were made using the index and middle fingers and the system was not allowed to receive more than one command within three seconds. To turn the screen on/off, the user needed to bring the index and middle fingers together while staying in the defined region of interest. The play/stop the music command was sent when the distance between two fingertips was more than 40 pixels. Moving the middle finger forward was assigned the command to play the previous song, and moving the index finger forward was assigned the command to play the next song. The experimental result showed that the algorithm was able to control the four functions on the smart phone.
Figure 5.8: (a) is the gesture for turning on/off the screen. (b) is the gesture for playing/stoping the music. (c) is the gesture for selecting the previous song. (d) is the gesture for the selecting next song.
Figure 5.9: The two fingers had the same depth. (a) and (b) are the background subtracted left and right image. (b) is the disparity map. The circles in (c) represent the depth of the fingertips.
Figure 5.10: The middle in the image moved forward. (a) and (b) are the background subtracted left and right image. (b) is the disparity map. The circles in (c) represent the depth of the fingertips.
6.1 Conclusion

In this thesis, we presented a robust stereo vision algorithm to obtain 3D information for human-computer interface applications. This algorithm achieved real-time performance without any manual optimization. Unlike the depth or 3D camera, which uses infrared to measure the distance and is easily affected by sunlight, this algorithm determines the object distance based on relative changes between pixels. It only requires very simple computation processes. It is suitable for applications that use resource-limited systems in an environment that contains sunlight.

Similar to the original version, the modified algorithm does not need the input image pair to be rectified or the stereo camera system to be in perfect canonical form. It also satisfies the requirement for human-computer interface purposes, which is that the algorithm is able to generate real-time and robust results for gesture analysis. The experimental results from human gesture estimation demonstrate the feasibility of controlling a consumer entertainment system like smart television or replacing a video game console. The experimental results from finger tracking show that this algorithm can be used to control a hand-held device for safety purposes.

6.1.1 Contributions

The original profile shape matching algorithm matches the vertices found in the row intensity profiles. However, some non-vertex points might be mistreated as vertices and some vertices might be treated as non-vertex points because of the noise. In such situations, the original algorithm would not find the best matches for the selected vertices. In order to find the best matches for every selected point, we must compare the selected points in the disparity range with every point in the other image. Although the modified algorithm requires slightly more computations than the original version, it actually needs fewer operations and hence has higher processing speed on the
tested stereo image pairs. This speed improvement is due to the smaller size of threshold set. We
demonstrated that the modified version provides 30% higher accuracy and approximately a 15%
 improvement on processing speed. Results presented in Section 5.4 show the robustness of the
proposed algorithm even when a small Gaussian noise is added to one of the input images.

6.2 Future Work

Improving accuracy while maintaining or even improving the processing speed remains a
challenge in stereo vision research. The new algorithm presented in this thesis could be improved
by developing a more efficient method to locate shape discontinuities that bound the shapes in the
intensity profile, or by using images containing more information such as multiple color channel
images, instead of using a single gray scale image. Designing and building an embedded vision
sensor for the proposed algorithm could be an important step to putting this work to practical use.
REFERENCES


56


