International Journal of Advanced Robotic Systems

OPEN ACCESS
ARTICLE

A New Profile Shape Matching Stereovision Algorithm for Real-time Human Pose and Hand Gesture Recognition

Regular Paper

Dong Zhang[1], Dah-Jye Lee[2,*] and Yung-Ping Chang[2]

1 School of Information Science and Technology, Sun Yat-Sen University Guangzhou, China
2 Department of Electrical and Computer Engineering, Brigham Young University, Provo, UT, USA
* Corresponding author E-mail: djlee@byu.edu

Received 31 Oct 2013; Accepted 08 Dec 2013

DOI: 10.5772/57515

Abstract This paper presents a new profile shape matching stereovision algorithm that is designed to extract 3D information in real time. This algorithm obtains 3D information by matching profile intensity shapes of each corresponding row of the stereo image pair. It detects the corresponding matching patterns of the intensity profile rather than the intensity values of individual pixels or pixels in a small neighbourhood. This approach reduces the effect of the intensity and colour variations caused by lighting differences. As with all real-time vision algorithms, there is always a trade-off between accuracy and processing speed. This algorithm achieves a balance between the two to produce accurate results for real-time applications. To demonstrate its performance, the proposed algorithm is tested for human pose and hand gesture recognition to control a smart phone and an entertainment system.

Keywords Profile Shape Matching, Real-Time Stereovision, Human Pose and Gesture Estimation, Resource-Limited Systems

1. Introduction

Stereovision has been the focus of computer vision research for several decades. Many new algorithms have been proposed in efforts to obtain better accuracy or higher processing speed. Generally, stereovision algorithms can be classified into sparse and dense algorithms [1]. Algorithms that output a sparse disparity map are usually feature-based methods and can produce accurate 3D information with limited density, which is not suitable for applications requiring dense-disparity information. As computational power has increased, dense-disparity methods have become popular in recent years and benefit more contemporary applications that demand dense output [1].

Scharstein and Szeliski proposed a method and infrastructure for quantitative evaluation of the accuracy of a dense stereovision algorithm [2]. This evaluation method provides a precise single number to determine the accuracy and has helped motivate researchers to

develop more accurate stereovision algorithms [3]. López-Valles [4] proposed calculation of the depth of the moving elements present in a video sequence by studying the correspondence of right and left image objects with a similar motion history, and achieved promising results. However, better accuracy almost always means more complicated computations. One stereovision algorithm may obtain highly accurate results, but the accompanying increased computational complexity limits its use for applications that require real-time performance.

To meet the requirements of many contemporary applications, e.g., autonomous robotic platforms, Simultaneous Localization and Mapping (SLAM), or Human-Computer Interaction (HCI), some researchers focus on improving and optimizing the existing algorithms to generate 3D information in real time. Many of these attempts rely on hardware that is able to run processes in parallel, such as multi-core systems and single-instruction multiple data (SIMD) instructions. These approaches take advantage of the parallelizable nature of stereovision algorithms to achieve good results in real time. Other developed algorithms for embedded systems include Field-Programmable Gate Array (FPGA), Digital Signal Processor (DSP), Application-Specific Integrated Circuit (ASIC), and low-power multi-core processors. Most of these have successfully accelerated the processing speed for stereovision applications.

With the wide use of light-weight, low-power, and passive-vision sensors, many stereovision applications require the algorithm to be implemented on resource-limited systems including mobile phones, entertainment game consoles, security systems, and unmanned aerial vehicles. Darabiha et al. [5] implemented a stereo depth measurement algorithm on a reconfigurable board with four Xilinx Virtex2000E FPGAs. Gehrig et al. [6] proposed a real-time implementation of a semi-global algorithm for automotive applications on a power-limited reconfigurable hardware platform with a Xilinx Virtex-4 FPGA. Tippetts et al. [7] implemented vision algorithms on an FPGA to provide additional information to supplement the insufficient data provided by a standard inertial measurement unit for micro-unmanned aerial vehicles. Samarawickrama implemented vision algorithms on FPGAs and evaluated their performance [8]. These systems usually have limited computational resources, e.g., constraints on weight, size, power, memory, and cost, which pose a big challenge to stereovision algorithm development due to computational complexity. Although advances in processing hardware technologies have been able to provide increasing computational power for systems with limited resources, studies on the trade-off between the accuracy and required resources continue to make contributions in advancing stereovision technology.

A new profile shape matching stereovision algorithm that uses simple computations and little computational resources is presented in this paper. The proposed algorithm extracts image intensity values from the same image rows of the stereo image pair and matches the intensity profile shape row by row. This approach reduces the effect of intensity and colour differences caused by lighting variations [3]. The algorithm produces a dense disparity map after it iterates through each row of the images. Once the disparity map is generated, a disparity range can be selected to remove objects and background features that are out of the desired distance range in front of the cameras. In other words, object detection is not based on an object's colour, intensity, or texture, but is determined by its distance from the cameras. Applications such as obstacle detection, human gesture recognition, and motion estimation can all benefit from isolating objects of interest from the background.

The proposed algorithm can be implemented in an embedded system, which requires simple computation processes for real-time performance and is valuable for resource-limited systems that require 3D information. This algorithm is proved to be capable of performing human pose and hand gesture recognition in real time. We present our literature review in Section 2. The proposed algorithm is explained in detail in Section 3. Experiment results on human pose and hand gesture estimation are discussed in Section 4. A conclusion is given in Section 5.

2. Background

2.1 Stereo vision

According to the way they assign disparities to pixels, dense disparity stereo matching algorithms can be categorized into local and global methods. Global methods assign a disparity value to each pixel according to the information on the whole image. Global methods are able to produce very accurate results but are time consuming and computationally demanding. Local methods determine the disparity of each pixel depending on the information on its neighbouring pixels. They are usually fast and able to produce decent results. These characteristics enable local methods to extract disparity map from image pairs in real time. Although global algorithms have produced the most accurate disparity maps in the past, many local algorithms have been developed in recent years that achieve competitive accuracy [9].

There have been significant efforts to increase the runtime performance of global techniques by reducing iterations or simplifying energy functions. For example, Bleyer and colleagues have developed algorithms that minimize global energy functions, such as Segm+visib [10],

WarpMat [11], and PatchMatch [12]. However, global methods are not suitable for real-time implementation, especially for resource-limited systems, because of their slow processing speed. Most existing real-time and near real-time stereovision algorithms are local methods.

Stereovision processing speed is often measured in million disparity evaluations per second, or Mde/s. Zhang et al. [13] proposed an adaptive correlation window approach able to achieve 6.328 Mde/s on a Pentium IV 3.0 GHz processor. Kosov et al. [14] applied a full approximation scheme to a multi-level adaption technique to minimize an energy function in RealtimeVar. Tippetts et al. [3] proposed a profile shape matching algorithm using intensity gradients to group and match shapes for each disparity level. This algorithm is able to process 640×480 images with a disparity range of 25 at 33f/s on one core of a standard CPU.

Research work has also been conducted using FPGA, ASIC, and DSP to implement a stereovision algorithm for real-time applications. Darabiha et al. [5] presented an FPGA-based stereo depth measurement algorithm which generates 8-bit sub-pixel disparities on 256×360 images at 30 f/s. Gehrig et al. [15] proposed a real-time implementation of a stereo-matching algorithm which processes 680×400 images at 25 f/s. Ambrosch and Kubinger [16] provided a realization of a stereo-matching algorithm as an Intellectual Property core designed for the deployment in FPGAs and ASICs. Jin et al. [17] proposed an FPGA implementation of a Census-based stereovision algorithm able to process 640×480 frames at 230 f/s. Gong et al. [18] employed a GPU to accelerate their algorithm, which can produce dense disparity maps for the *Tsukuba* image pair at a rate of 16 f/s.

In our previous work [3], a stereovision algorithm based on intensity profile shape matching was proposed for 3D human gesture analysis and obstacle detection. This algorithm does not require complex computations and is sufficient for certain tasks that require 3-D information. In this paper, we present an improved version for real-time applications for resource-limited systems. Comparison results show the proposed algorithm obtains higher accuracy and performs 1.14 times faster than the original method in [3].

2.2 Human gesture and motion estimation

Computer-vision-based human gesture and motion estimation techniques are capable of providing the user with a more natural and intuitive way to communicate with a computer by using a set of video cameras and computing hardware to interpret human gestures and finger motion. The non-obstructive nature of the resulting vision-based interface has led to a burst of recent activities in this research area [19].

Gavrila [20] proposed grouping human motion analysis methods into 2D and 3D approaches. Aggarwal and Cai [21] classified the research in this field into three categories: body structure analysis, tracking, and recognition. The essential part of body structure analysis is pose estimation, which can be split into model-based and model-free, depending upon whether a priori information about the object shape is provided. Wang and Singh [22] separated the process of computational analysis of human movement into a tracking phase and a motion analysis phase. Tracking is performed for hands, heads and full bodies. Poppe [23] reviewed the taxonomy of vision-based human motion analysis. Turaga et al. [24] provided an overview of methods to perform machine recognition of human activities.

In this paper, we use a new profile shape matching stereovision algorithm for real-time human pose and hand gesture recognition to control a smart phone or a hand-held device. Testing results demonstrate that the proposed stereovision algorithm achieves a balance between the accuracy and processing speed to produce accurate results in real time for human pose and hand gesture recognition.

3. Profile shape matching algorithm

3.1 Algorithm overview

Figure 1 shows a Gaussian-filtered *Tsukuba* stereo image pair. Figure 2 shows the Gaussian-filtered row intensities from the *Tsukuba* stereo image pair in Figure 1(a) and Figure 1(b). It can be observed that the profiles of the two rows share a similar pattern that can be used to determine their disparity. Upon further examination, one profile actually contains spans that are not found in the other [3]. Examples of these subtle differences representing discontinuities in the disparity map are highlighted in Figure 3. The proposed algorithm assumes that all points included in an intensity profile shape, bounded by discontinuities, correspond to the same depth plane. Similar assumptions are found in [25-27]. It also assumes that a point can only match one unique corresponding point in the other image. This is known as the uniqueness constraint. The new stereovision algorithm developed in this work is based on these unique assumptions.

Figure 1. Gaussian-filtered *Tsukuba* stereo image pair; (a) is the left image and (b) is the right image

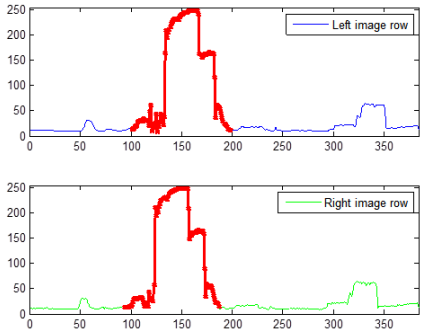


Figure 2. Row profiles extracted along the white line in Figure 1(a) and Figure 1(b)

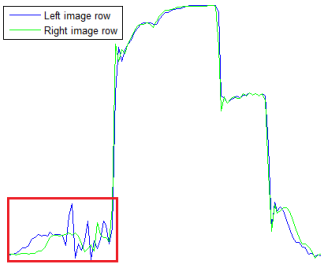


Figure 3. Red rectangle highlights the discontinuity between two superimposed profiles

The proposed algorithm consists of five steps (see Figure 4). A Gaussian filter is applied to both the left and the right images. The purpose of this step is to reduce noise and smooth the intensity profile to increase profile shape matching accuracy. Then the proposed algorithm selects candidate points for matching in the left image, based on the intensity changes between two neighbouring pixels. A voting mechanism is employed to identify the corresponding point in the right image for each candidate point. After a process of profile shape growing, all pixels are assigned a shape label. The final process of vertical smoothing is performed to include the information contained in the image columns.

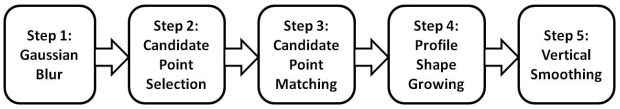


Figure 4. Overview of the proposed algorithm

3.2 Candidate point selection

After the images are smoothed, the algorithm selects points to match based on the intensity changes between two neighbouring pixels. The selected points are called candidate points. There are two requirements for a pixel to be selected as a candidate point. First, the absolute intensity difference between the respective point and the next pixel is calculated. The absolute intensity difference will only be added to the accumulated change if it is greater than or equal to the minimum step threshold (MST). Second, a pixel will only be selected as a candidate point when its accumulated intensity difference is greater than or equal to the accumulated change threshold (ACT).

Figure 5 shows a couple of examples of the candidate point selection process where the MST and ACT are set to 2 and 5, respectively. Starting from the left pixel, the absolute intensity difference between the two adjacent pixels is 4. It is added to the total change (initialized to 0) because it is greater than 2 (the MST). The second pixel has an absolute intensity difference of 1, so it is ignored. The third pixel has an absolute intensity difference of 2, so it is added to the total change to make 6 (4+2=6). The third pixel is selected as a candidate point because the total change is greater than 5 (the ACT). Once a candidate point is selected, the process repeats from the next point with the total accumulative change reset to 0. The absolute intensity differences (arrows) highlighted in red are ignored because they are smaller than the MST. The two circled points are selected as candidate points because their total changes are greater than the ACT. This selection process reduces the possibility of selecting candidate points from flat surfaces and selects more candidate points from surfaces with abundant variances.

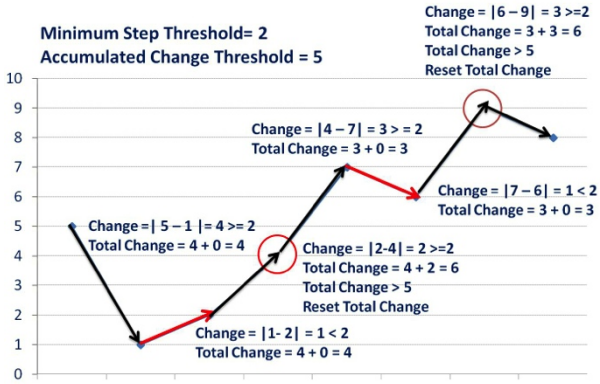


Figure 5. Example of the candidate point selection process with MST set to 2 and ACT set to 5

3.3 Candidate point matching

For each candidate point in the left image, its corresponding candidate point in the right image is identified with a voting mechanism in Step 3. Every candidate point in the left image is treated as a starting point for searching and is expanded over a span of a certain length, l. The slope S_x^a between two consecutive pixel points is defined in Equation (1):

$$S_x^a = I_x^a - I_{x-1}^a \quad (1)$$

where x is the row pixel index, I_x^a is the intensity value of the point indexed by x, 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, and repeat the same process for every pixel point in the right image which lies within the desired range, i.e., $\left[P_i^L - LB, P_i^L - UB\right]$, where P_i^L is the pixel index of the i-th candidate point of the left image. The proposed algorithm compares the slope S_y^R around the points within the desired range of the right image against the slope S_x^L around the candidate points of the left image.

Two parameters, f and f^{zc}, are defined for candidate point matching. If the two compared slopes have the same sign and their absolute difference is less than the Same-Sign Threshold f, then they are considered matching patterns and the vote count is incremented. When S_x^L and S_y^R have different signs but their absolute difference is less than the Zero-Crossing Threshold, f^{zc}, the two slopes can also be regarded as matching patterns and the vote count is incremented. The value of f^{zc} is positive and less than f. The point of a span with the largest count of vote is selected as corresponding to the candidate point. This candidate point matching process is summarized in Algorithm 1.

for $i = 1$ to number of selected candidate 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 $x = P_i^L - l, y = P_j^R - l$ to $x = P_i^L + l, y = P_j^R + l$ **do**
      if S_x^L and S_y^R have the same sign and $\left|S_x^L - S_y^R\right| < f$ **then**
        Increment $vote$
      end
      if S_x^L and S_y^R have different signs and $\left|S_x^L - S_y^R\right| < f^{zc}$ 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

Algorithm 1. Looking for corresponding points in the right image for each row

The employed voting mechanism finds matching patterns according to the gradient rather than the intensity value. It also ensures the slopes have similar orientation by testing their signs. This unique voting mechanism is robust even if the input image pair includes noise.

An example of finding matching candidate points is shown in Figure 6. The same profile shown in Figure 5 is used as the left profile. In this example, the Zero-Crossing Threshold and the Same-Sign Threshold are set to 1 and 2, respectively. The algorithm starts by comparing the two circled points in Figure 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 the right profiles is less than the Same-Sign Threshold (set to 2). As a result, it receives one 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 the penultimate point, as shown in Figure 6(b). The differences of the left and right slopes are both less than the Same-Sign Threshold, and the vote is the highest in the right profile. Therefore, these are considered the best match. The disparity value of the mth point pair is defined as the difference of their row pixel indices.

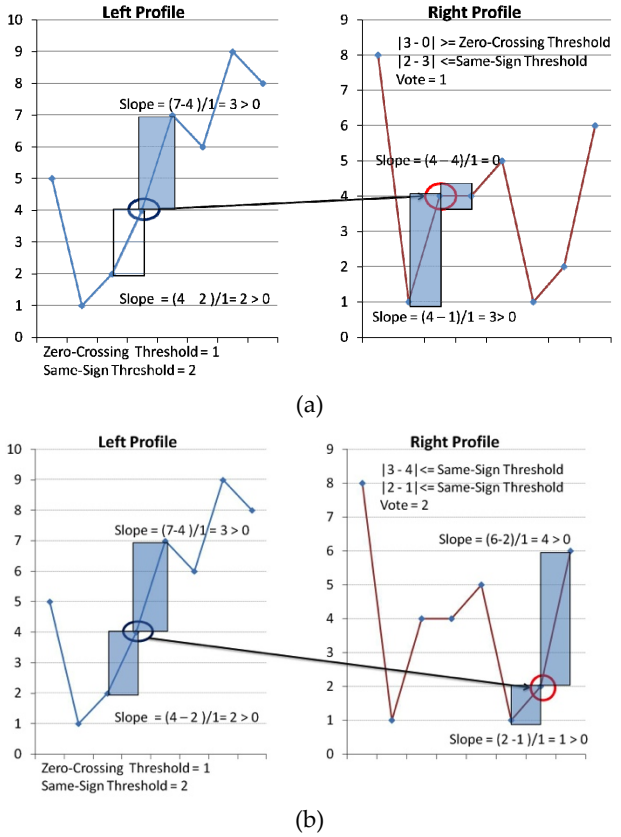


Figure 6. An example of finding the corresponding point in the right profile with a Zero-Crossing Threshold of 1 and a Same-Sign Threshold of 2; (a) receive one vote; (b) receive two votes

3.4 Profile shape growing

Although candidate points are matched in the previous step, the majority of pixels of the current image row are still not included in any pattern. In Step 4, the patterns in the current row will expand from the matched points to cover every pixel using an increasing threshold. This process is summarized in Algorithm 2.

Let q_m^a and p_m^a denote the beginning index and the end index of the span expanded from the mth point pair, respectively, and say a can be L (left image) and R (right image). The slopes at q_m^a and p_m^a can be obtained with Equation (1), and denoted as $S_{q_m}^a$ and $S_{p_m}^a$, respectively. We also define two threshold sets, T and T^{zc}, for the matching process. The elements of these two threshold sets, $t_n \in T$ and $t_n^{zc} \in T^{zc}$, are increased in iterations.

The absolute difference between $S_{q_m}^L$ and $S_{q_m}^R$ (or $S_{p_m}^L$ and $S_{p_m}^R$) is compared against the thresholds in each iteration. When one of the slopes of $S_{q_m}^L$ and $S_{q_m}^R$ (or $S_{p_m}^L$ and $S_{p_m}^R$) 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 neighbouring points. If $S_{q_m}^L$ and $S_{q_m}^R$ (or $S_{p_m}^L$ and $S_{p_m}^R$) have different signs, the threshold t_n^{zc} is used to evaluated the difference. The following list shows all the conditions under which the neighbouring points of q_m^L and q_m^R (or p_m^L and p_m^R) can be considered to be part of the same pattern:

(c-1) One of the slopes of $S_{q_m}^L$ and $S_{q_m}^R$ (or $S_{p_m}^L$ and $S_{p_m}^R$) is zero;

(c-2) $S_{q_m}^L$ and $S_{q_m}^R$ (or $S_{p_m}^L$ and $S_{p_m}^R$) have the same sign and their difference is less than t_n;

(c-3) $S_{q_m}^L$ and $S_{q_m}^R$ (or $S_{p_m}^L$ and $S_{p_m}^R$) have different signs, but the absolute difference is less than t_n^{zc}.

The values of t_n^{zc} and t_n are set by the user to make the algorithm robust to noise. Because a slope pair with the same sign is more likely from the same pattern, while a slope pair with different signs is less likely from the same pattern, we set the value of t_n^{zc} to be smaller than the value of t_n.

A similar process takes place for $S_{p_m}^L$ and $S_{p_m}^R$. This process only assigns a shape to the points that have not been included into any shape. Thus, the span $\left[q_m^a, p_m^a\right]$ grows longer with the increasing of the thresholds until all pixels are assigned with a shape label.

```
for t_n in threshold set T do
    error = 0
    for m = 1 to number of matched point pairs do
        while error == 0 do
            if (c-1) or (c-2) or (c-3) is satisfied
                decrease q
            else
                error = 1
            end
        end
        error = 0
        while error == 0 do
            if (c-1) or (c-2) or (c-3) is satisfied
                increase p
            else
                error = 1
            end
        end
    end
end
```

Algorithm 2. Matching patterns for each row

3.5 Vertical smoothing

The final step of the proposed algorithm is a vertical smoothing process. Since the matching process is a row-wise operation, the information contained in the image columns is not considered in the previous steps of the process. The vertical smoothing process accumulates the five pixels above and below the target pixel according to the disparity value. The target pixel is then modified to take the disparity value with the greatest number of votes. This 11-pixel voting scheme produces the final disparity map.

3.6 Discussion

The proposed algorithm selects a candidate point when the accumulated change is greater than the accumulated change threshold. Every selected point expands a span with l pixel width. A total of $3\times(l-1)$ subtractions and $2\times(l-1)$ comparisons are needed for comparing slopes in two spans. Let $d_{\max}$ denote the desired disparity range, and $d_{\max}$ iterations are needed for determining a best match. In the worst-case scenario, we assume all differences between pixels are greater than the accumulated change threshold, i.e., every point is assumed to be a candidate point. The number of operations per pixel is $3\times(l-1)\times d_{\max}$ plus $2\times(l-1)+d_{\max}$ comparisons.

Compared with our previous version of the profile shape matching algorithm, presented in [3], this improved

version 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.

4. Testing and results

The new stereovision algorithm was first tested on a *Tsukuba* stereo image pair to evaluate its accuracy. It was also tested for human pose and hand gesture recognition to demonstrate its real-time performance. For human pose recognition, the disparity map obtained with the proposed algorithm was first used as a mask to isolate the human body from the background. Human pose was then estimated by analysing the relative 3D positions of the joints of a human skeleton model. For hand gesture recognition, after locating the fingertips, the depth and the relative finger positions in the region of interest were used to estimate the hand gesture to control a smart phone.

4.1 Algorithm parameters

A brute force method was applied to find the optimal combination of parameters for different applications. An optimal combination of parameters was selected to generate the disparity map with high accuracy and fast processing speed. Kernel sizes from 5×5 to 25×25 with standard deviation values from 1 to 10 with 0.5 intervals for Gaussian filter were tested. There are a total of seven parameters from Step 2 to Step 4 of the proposed algorithm: MST, ACT, l, f, f^{zc}, t_n, and t_n^{zc}. The brute force approach was used to determine the best combination of these seven parameters. The combination of parameters that generates the best result in *Tsukuba* and human pose estimation was a 9×9 kernel size with a standard deviation value of 3; the rest of the parameters are listed in Table 1.

	Tsukuba	Human pose estimation
MST	5	0.5
ACT	30	10
l	5	10
f	1	5
f^{zc}	0.4	0
t_n	1, 5, 8	0.4, 0.8, 1.5, 2
t_n^{zc}	0.4	0.3, 0.4

Table 1. The parameters that generate the best result in *Tsukuba* and human pose estimation

4.2 Hardware

The profile shape matching algorithm was implemented in C++ without any manual code optimization and executed on a machine with an AMD Phenom II 2.8 GHz processor. The stereo camera system for human gesture estimation is shown in Figure 7. This system uses two PointGrey Flea®2 CCD cameras. The two cameras were adjusted manually so that their optical axes were close to parallel. The images were used in our algorithm without rectification to prove the algorithm's robustness. The resolution of both input images was 640×480. The Gaussian filter, morphological operation, Haar face detection, flood fill algorithm, linear regression, and convex hull algorithm were implemented using the OpenCV library.

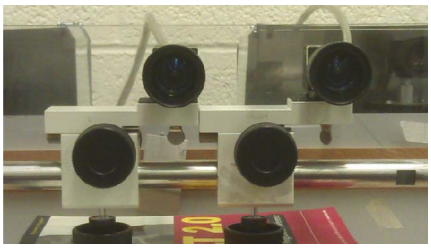


Figure 7. The stereo camera system for human pose estimation

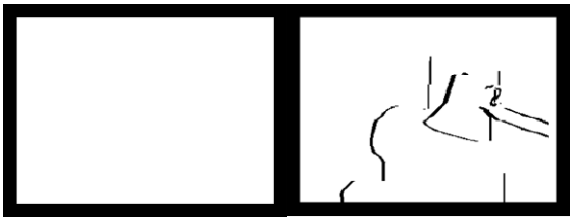
4.3 Accuracy measurement

The metric used for accuracy measurement is called bad matching pixel measurement [2]. This computes the disparity difference between the obtained disparity map and the ground truth in the same pixel index. A pixel is regarded as a bad matching pixel if its disparity error is greater than a threshold, δ_d, which is called the disparity error tolerance and set by the user to evaluate the quality of the computed correspondences [2]. The percentage of bad matching pixels is calculated as

$$B = \frac{1}{N}\sum_{x,y}\left(\left|d_C(x,y) - d_T(x,y)\right| > \delta_d\right) \tag{2}$$

where N is the total number of pixels, $d_C(x,y)$ is the value of a pixel point in the obtained disparity map, and $d_T(x,y)$ is the disparity value of a pixel point 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 8(a). The second case computes the difference of the non-occluded region, which is shown as white areas in Figure 8(b). The occluded areas highlighted in black are excluded. The ground truth for both cases is obtained from the Middlebury online library.

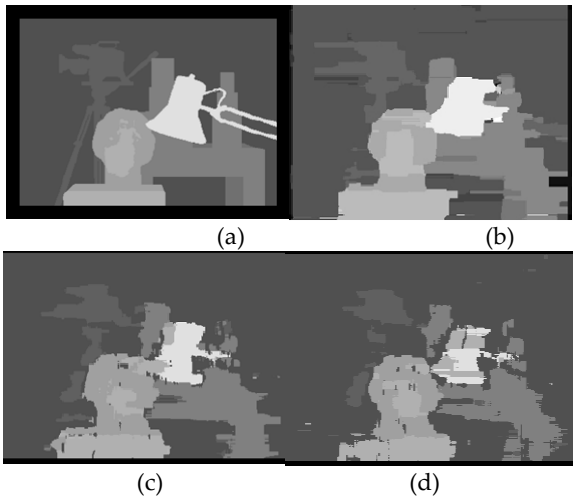


(a) (b)

Figure 8. Bad matching pixels are computed in the whole image in (a) and in the non-occluded region in (b)

4.4 Resulting images of Tsukuba stereo image pair

The *Tsukuba* stereo image pair is a well-known computer generated image pair for testing the accuracy of stereovision algorithms. Figure 9(a) shows the ground truth of its disparity map. Different intensity values represent different distances from the object to the stereo camera system. Figure 9(b) is the result produced by the previous version [3]. Figure 9(c) is the disparity map obtained with the improved profile shape matching algorithm. Figure 9(d) demonstrates the disparity map when a zero mean Gaussian noise with standard deviation of 2 was added to the right image. The result demonstrates the more-than-adequate accuracy and robustness of the proposed algorithm.



(a) (b)

(c) (d)

Figure 9. (a) The ground truth of the *Tsukuba* stereo image pair. (b) The disparity map of the original algorithm. (c) The disparity map of our algorithm. (d) The disparity map obtained by our algorithm when a Gaussian noise with zero mean and a standard deviation of 2 is added to the right image.

	Images	*nonocc*≥1	*all*≥1	*nonocc*>2	*all*>2
Original Version	Tsukuba	9.6	11.5	3.2	5.0
Improved Version	Tsukuba	6.5	8.1	2.2	3.6

Table 2. Percentage of bad matching pixels in the disparity map

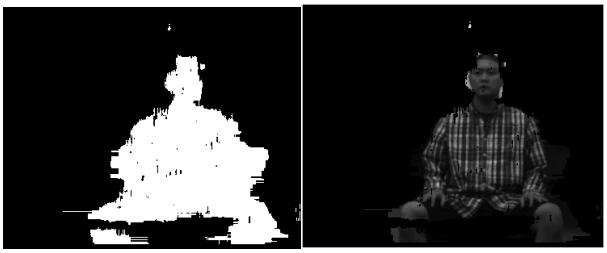
The percentages of bad matching pixels given by the original algorithm and this improved version are shown in Table 2. The accuracy measurement was calculated using Equation (2). The columns with the label *all* show the differences between the ground truth and the obtained disparity map for every pixel except those in the black region on the border, as shown in Figure 8(a). In the case of *nonocc*, the accuracy measurement calculates the disparity differences pixel by pixel only in the white areas in Figure 8(b).

The labels *nonocc*≥1 and *all*≥1 indicate that a pixel is counted as a bad matching pixel when its disparity difference is greater than or equal to 1. The labels *nonocc*>2 and *all*>2 represent that a pixel is considered as a bad matching pixel when the disparity difference is greater than 2. In all cases, the proposed algorithm has a lower percentage of bad matching pixels than the original version. The improved version removes approximately 30% more bad matching pixels than the original version. The average processing time for the entire profile shape matching algorithm, from Gaussian smoothing to vertical smoothing, was 16 mSec (62 fps); it is now 14 mSec (66 fps) for the improved version. The improved version is approximately 1.14 times faster than the version presented in [3]. A performance comparison with other stereovision methods is already included in [3] and will not repeated here.

4.5 Human pose estimation

Using the disparity map obtained by the proposed algorithm as a mask, objects within the desired distance range are detected and retained. Objects outside the range are removed. In this application, the region of the body was segmented from the background. A face detection algorithm [28] was used to localize the blob of the body. Once the coordinate of the face was obtained, a flood fill algorithm was applied to find the connected components of the face. Using the labelled blob of the body and the size of the detected face, a human skeleton model was built for pose estimation. The ratios between human joints were fixed and determined based on the NASA Anthropometric Source Book [29]. Then the disparity values were assigned to the corresponding joints of the skeleton model. The pose was then estimated with the depths and the relative positions of the joints.

Figure 10 shows the created mask and the segmented region of the body. Figure 11 shows the result of the flood fill algorithm using the face coordinate as the seed, and an example of human skeleton fitting.



(a) (b)

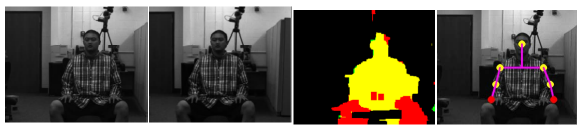
Figure 10. The created mask (a) and the segmented region of the body (b)

(a) (b)

Figure 11. The result of the flood fill algorithm using the face coordinate as the seed (the blue region indicates the blob of the body) (a) and an example of skeleton fitting (b)

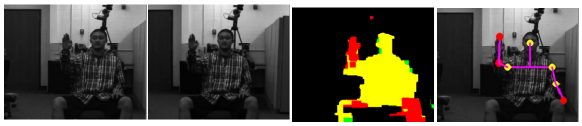
In Figure 11, the disparity range was selected at between 35 to 45 pixels. The background was removed because it was outside the desired distance range. Different colours were used to represent the object distances: green was for the farthest objects, yellow for the mid-range, and red for the closest. In order to speed up the process, we specified a certain region for face detection. As a result, the process only worked when a face appeared in the region.

Figures 12(a) and 12(b) show the left and right input images from the stereo camera system, respectively. The human model put both hands on his knees, as shown in red in Figure 12(c). The red colour indicates that the hands were closer to the cameras than the green parts of the body. The colour dots on the human skeleton model in Figure 12(d) represent the depths of each part. Figures 13(a) and 13(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 colour (Figures 13(c) and 13(d)). The images for the right arm moved backward are shown in Figures 14(a) and 14(b). The movement was detected as shown in Figure 14(c), and the obtained skeleton model is shown in Figure 14(d).



(a) (b) (c) (d)

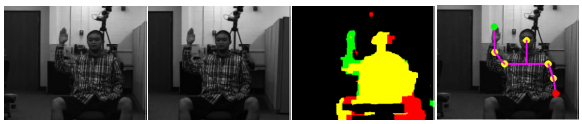
Figure 12. The left (a) and the right (b) input images; the generated human model (c); the human skeleton model with colour dots reflecting the depth of each joint (d)



(a) (b) (c) (d)

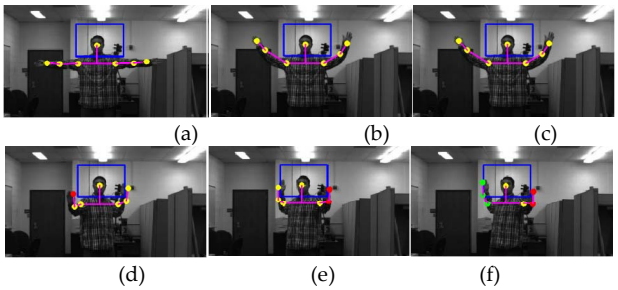
Figure 13. The left image (a) and the right image (b) for right arm raised and moved forward; the generated human model (c); the human skeleton model with colour dots reflecting the depth of each joint (d)

More results are shown in Figure 15. The algorithm detects the face in the blue rectangle. The colour of the joints in Figures 15(a)-(c) indicates they were at the same distance from the cameras. The red dot in Figure 15(d) in the right-hand position shows that the hand was closer to the cameras than other joints. A similar result can be seen in Figure 15(e), which shows the left arm moved forward. In Figure 15(f), the green dots on the right arm indicate that they were farther from the cameras than the head. By analysing the colour dots in Figure 15, the human gesture can be determined and used for interfacing with a computing device.



(a) (b) (c) (d)

Figure 14. The left image (a) and the right image (b) for right arm raised and moved backward; the generated human model (c); the human skeleton model with colour dots reflecting the depth of each joint (d)



(a) (b) (c)

(d) (e) (f)

Figure 15. The skeleton fitting results of different gestures. All joints in (a), (b), and (c) were on the same depth plane. The right hand in (d) was moved forward. The left arm was moved forward in (e). In (f), the left arm was moved forward and the right arm moved backward.

4.6 Hand gesture estimation

For hand gesture estimation, similarly to human pose estimation, the background was removed by retaining objects within the desired distance range and excluding those outside the range. The hand region was then extracted from the background and the convex hull of the hand was determined by using the method presented in [30]. Since the vertices of the convex hull may not always be formed by fingertips, we selected a region of interest and only detected the vertices in this region. An example is shown in Figure 16. All five fingertips are depicted by red circles.

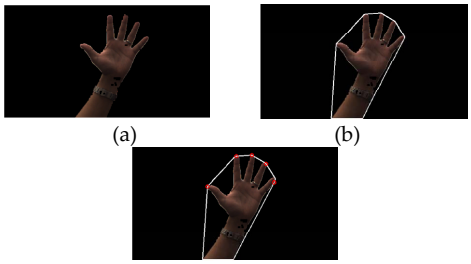


Figure 16. Segmented hand from the background (a); the convex hull of the hand (b); the fingertips forming the convex hull (c)

According to the obtained disparity map, disparity values were assigned to these fingertip locations. The depth and the relative position of the fingertips in the detecting region were utilized to estimate hand gesture. Each estimated gesture was employed as a specific function to control a smart phone or tablet computer. The analysed results are interpreted as specific keys on the keyboard. An Android application, ShareKM [31], which maps the keyboard inputs to the software functions, was used to control a hand-held device.

Figures 17(a) and (b) show the regions of the hand which were segmented from the left and right input images. In Figure 17, the index and middle finger had the same disparity value as the palm and arm. This is reflected in Figure 17(c). The hand gesture estimation was obtained by combining the disparity values with the locations of the fingertips. Figure 17(d) shows the estimation result. The tips of the index and middle fingers had the same distance and were assigned the same colour.

The middle finger was moved forward in Figures 18(a) and 18(b). As a result, the middle fingertip was assigned yellow, as shown in Figure 18(c). The depths of the fingertips are shown in different colours in Figure 18(d). Depending on the distance and the depth of the fingertips, different hand gestures were recognized. Since each defined gesture had been assigned a key input of a keyboard, through the Android application mentioned previously, the keys were mapped to the defined functions to control a smart phone.

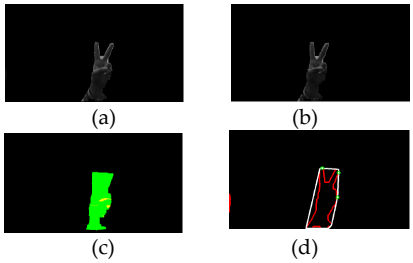


Figure 17. The hand regions segmented from the left (a) and right (b) input images; the disparity map of the hand (c), where the index and middle fingers had the same disparity value as the palm and arm; the estimation of the hand gesture (d)

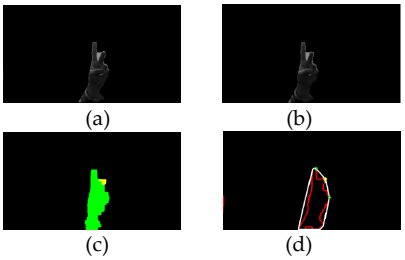


Figure 18. The hand regions segmented from the left (a) and right (b) input images, where the middle finger was moved forward; the disparity map of the hand (c); the estimation of the hand gesture (d)

In our experiment for human hand gesture estimation, the region of interest was a 200×100 area in the middle of the image; the colour representations for distances were the same as mentioned in Section 4.5. We assigned four gestures, as shown in Figure 19, to control a smart phone: turn on/off screen, play/stop music, play previous song, and play next song. Commands were given using the index and middle fingers. The system was not allowed to receive more than one command within three seconds.

To turn on/off the screen, the user needed to bring the index and middle fingers together while staying in the defined region of interest. The play/stop music command was sent when the distance between two fingertips was more than 40 pixels. Moving the middle finger forward was assigned to play the next song, and moving the index forward was assigned to play the previous song. The experiment results showed that the algorithm was able to control the four functions on a smart phone.

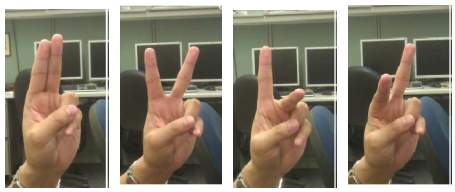


Figure 19. Defined hand gestures: (a) turn on/off; (b) play/stop music; (c) select the previous song; (d) select the next song

5. Conclusion

This paper presents a new stereovision algorithm that is efficient for real-time applications. This algorithm achieves 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 lighting variations, this algorithm determines the object distance based on relative intensity changes. It requires very simple computations. It is suitable for applications that use resource-limited systems in either indoor or outdoor environments.

The proposed algorithm does not require the input image pair to be rectified or the stereo camera system to be in perfect canonical form. It is able to generate an accurate disparity map for human gesture analysis. Although the performance of this algorithm for the two applications discussed in Sections 4.5 and 4.6 is hard to quantify, the results shown in Figures 12-19 are all quite good. Occasionally, there were a very few frames where the joint or fingertip locations were not quite accurate. These intermittent errors did not affect the final post and gesture estimations. The experimental results demonstrate the accuracy and robustness of the proposed algorithm and show its feasibility for real-time human gesture estimation for video-based non-contact human-computer interfaces.

6. Acknowledgements

The authors gratefully acknowledge the financial support of the National Science Foundation of China (No.61100170) and the Fundamental Research Funds for the Central Universities of China (No.12lgpy37).

7. References

[1] Lazaros N, Sirakoulis G C, Gasteratos A (2008) Review of stereo vision algorithms: From software to hardware. International Journal of Optomechatronics. J. 2: 435-462.

[2] Scharstein D, Szeliski R (2002) A taxonomy and evaluation of dense two-frame stereo correspondence algorithm. Int. J. Comput. Vision. J. 47(1-3): 7-42.

[3] Tippetts B J, Lee D J, Archibald J K, Lillywhite K D (2011) Dense disparity real-time stereo vision algorithm for resource-limited systems. IEEE Transactions on Circuits and Systems for Video Technology. J. 21(10): 1547 -1555.

[4] López-Valles J M, Fernández M A, Fernández-Caballero A (2007) Stereovision depth analysis by two-dimensional motion charge memories. Pattern Recognition Letters. J. 28: 20-30.

[5] Darabiha A, Rose J, Maclean J (2003) Video-rate stereo depth measurement on programmable hardware. In: Proceedings of 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1: 203-210.

[6] Gehrig S K, Eberli F, Meyer T (2009) A real-time low-power stereo vision engine using semi-global matching. In: Proceedings of the 7th International Conference on Computer Vision Systems. Ser. Lecture Notes in Computer Science. (5815):134-143.

[7] Tippetts B J, Lee D J, Fowers S G, Archibald J K (2009) Real-time vision sensor for an autonomous hovering micro unmanned aerial vehicle. Journal of Aerospace Computing, Information, and Communication. J. 6(10): 570-584.

[8] Samarawickrama M G (2010) Performance evaluation of vision algorithms on FPGA. Master's thesis, Dept. Electron. Telecommun. Eng., Univ. Moratuwa, Moratuwa, Sri Lanka.

[9] Tippetts B., Lee D J, Lillywhite K, Archibald J (2012) Efficient stereo vision algorithms for resource-limited systems. Journal of Real-Time Image Processing, Sept. 2012.. Doi: 10.1007/s11554-012-0268-3.

[10] Bleyer M, Gelautz M (2005) A layered stereo matching algorithm using image segmentation and global visibility constraints. ISPRS J. Photogramm. Remote Sens. J. 59(3): 128-150.

[11] Bleyer M, Gelautz M, Rother C, Rhemann C (2009) A stereo approach that handles the matting problem via image warping. In: IEEE Conference on Computer Vision and Pattern Recognition, 2009 (CVPR 2009): 501-508.

[12] Bleyer M, Rhemann C, Rother C (2011) Patchmatch stereo-stereo matching with slanted support windows. In: British Machine Vision Conference (BMVC): 1-11.

[13] Zhang K, Lu J, Lafruit G, Lauwereins R, Gool L V (2009) Real-time accurate stereo with bitwise fast voting on CUDA. In: 2009 IEEE 12th International Conference on Computer Vision Workshops (ICCV Workshops): 794-800.

[14] Kosov S, Thormahlen T, Seidel H (2009) Accurate real-time disparity estimation with variational methods. In: Proceedings of the 5th International Symposium on Advances in Visual Computing (ISVC): 796-807.

[15] Gehrig S, Franke U (2007) Improving sub-pixel accuracy for long range stereo. In: Proceedings of the IEEE 11th International Conference on Computer Vision: 1-7.

[16] Ambrosch K, Kubinger W (2010) Accurate hardware-based stereo vision. Computer Vision and Image Understanding. J. 114(11): 1303-1316.

[17] Jin S, Cho J, Pham X D, Lee K M, Park S K, Kim M, Jeon J W (2010) FPGA design and implementation of a real-time stereo vision system. IEEE Trans. Circuits Syst. Video Technology. J. 20(1): 15-26.

[18] Gong M, Yang Y H (2005) Near real-time reliable stereo matching using programmable graphics hardware. In: Proc. IEEE Comput. Soc. Conf. CVPR, 1: 924-931.

[19] Pavlovic V I, Sharma R, Huang T S (1997) Visual interpretation of hand gestures for human-computer interaction: A review. IEEE Transactions on Pattern Analysis and Machine Intelligence. J. 19(7): 677-695.

[20] Gavrila D M (1999). The visual analysis of human movement: A survey. Computer Vision and Image Understanding (CVIU). J. 73 (1): 82-92.

[21] Aggarwal J K, Cai Q (1999) Human motion analysis: A review. Computer Vision and Image Understanding (CVIU). J. 73(3): 428-440.

[22] Wang J, Singh S (2003) Video analysis of human dynamics: A survey. Real-Time Imaging. J. 9(5): 321-346.
[23] Poppe R (2006) Vision-based human motion analysis: An overview. Computer Vision and Image Understanding. J. 108: 4-18.
[24] Turaga P, Chellappa R, Subrahmaniman V S, Udrea O (2008) Machine recognition of human activities: A survey. IEEE Trans. Circuits Syst. Video. Technol. J. 18(11): 1473-1488.
[25] Klaus A, Sormann M, Karner K (2006) Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. In: Proceedings of the 8th International Conference on Pattern Recognition. 3: 15-18.
[26] Bobick A F, Intille S S (1999) Large occlusion stereo. International Journal of Computer Vision. J. 33: 181-200.
[27] Brockers R (2009) Cooperative stereo matching with colour-based adaptive local support. In: Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns. Ser. CAIP '09. Berlin, Heidelberg: Springer-Verlag, 1019-1027.
[28] Viola P, Jones M (2001) Rapid object detection using a boosted cascade of simple features. In: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), 1: 511-518.
[29] NASA (1978) Anthropometric Source Book, Springfield VA, Johnson Space Center, Houston, TX, 2.
[30] Graham R L (1972) An efficient algorithm for determining the convex hull of a finite planar set. Information Processing Letters. J. 1(4): 132-133.
[31] ShareKM. Available online: https://sites.google.com/site/droidskm/ Accessed: March 1, 2013