Automatic Human Joint Detection Using Microsoft Kinect

Samuel Cahyawijaya
Informatics Engineering
Institut Teknologi Bandung
Bandung, Indonesia
samuel.cahyawijaya@gmail.com

Iping Supriana Suwardi
Informatics Engineering
Institut Teknologi Bandung
Bandung, Indonesia
iping@informatika.org

***Abstract*—Automatic human joint detection has been used in many application nowadays. In this paper, we propose an approach to detect full body human joint method using depth and color image. The proposed solution is divided into 3 stage, which is image preprocess stage, distance transform stage, and anthropometric constraint analysis stage. The output of our solution is a stickman model with the same pose as in the given input image. Our implementation is done by using a Microsoft Kinect RGB and depth camera with 480x640 image resolution. The performance of this solution is demonstrated on several human posture**.

Keywords— Microsoft Kinect, human joint, anthropometry, depth image, body measure

I. INTRODUCTION

Human joint detection is an important task in computer vision. Many applications have implemented joint detection for various kind of task, such as human motion analysis, action recognition, surveillance, etc. There are many method has been implemented to do human joint detection, one is done by manually set the joint on the image which is a slow method to be performed. Other method uses optical marker on the subject which lead to expensive but faster method. Other method is done without using marker on RGB image, but there is an accuracy problem with this method.

Recently, due to emersion of low-cost depth camera sensor markerless approach of human joint detection using only depth camera or both RGB and depth camera has been developed. Some implementation is done by using anthropometric constraint and some other is done by using random forest classifier. In this paper, we propose an approach to detect single full body human joint detection by using a single Microsoft Kinect camera. Our solution divided into 3 stage which is preprocess stage, distance transform stage, and anthropometry analysis stage.

II. RELATED WORKS

Kar [4] and Jain et al.[3] have developed a novel human upper body joints detection method. Both use the same face detection ,distance transform, and anthropometric analysis. The different part is on the arm fitting method. On Kar implementation, a skin segmentation method is use to do the arm fitting, while on Jain et al. Implementation, arm fitting is done by sampling method on the weighted-distance transform result. (Kar, 2010) implementation use 640 x 480 RGB and depth image with average detection time of 103 ms, while Jain et al. Implementation use 320 x 240 RGB and depth image with average detection time of 68 ms.

Shotton et al.[2] propose a machine learning method to approximate human joints by detecting body part first. Shotton et al. use random forest to detect the body parts with 15.000 and 900.000 training data. The resulting experiment give average detection accuracy from 73.1% up to 98,4%.

Currently Microsoft Kinect SDK also provide joint detection method. The SDK also use random forest algorithm to detect human joints and given joint position has been converted to metric size. The Microsoft Kinect SDK also give method to convert the joint position into many resolution that supported by Microsoft Kinect..

III. SYSTEM OVERVIEW

All related works is done by using a camera with aspect ratio of 4:3. To get a full picture of standing pose human body with Microsoft Kinect we need to be at approximately 2 - 2.5 meters away from the camera. On our solution, we try to minimize the range of object to camera by rotating the camera to get aspect ratio of 3:4. By doing so, we could decrease the range of detection to approximately 1.6 - 2.1 meters away from the camera.

Our solution assumes that the camera position is static and the object captured by camera is an unobstructed full body of human. Our solution workflow is divided in four steps as shown in Fig. 1. The output from our solution is a stickman model consists of 21 joints. The joint structure is based on tree data structure as shown in Fig. 2

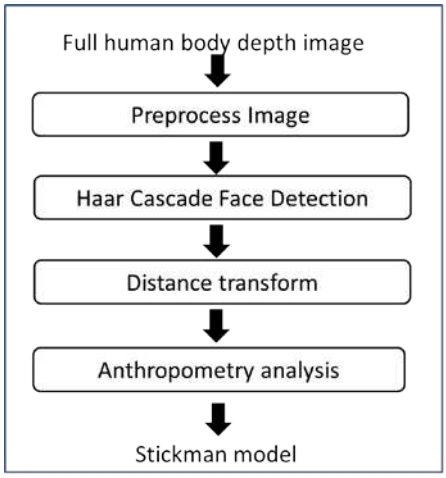

Fig. 1. Proposed solution workflow

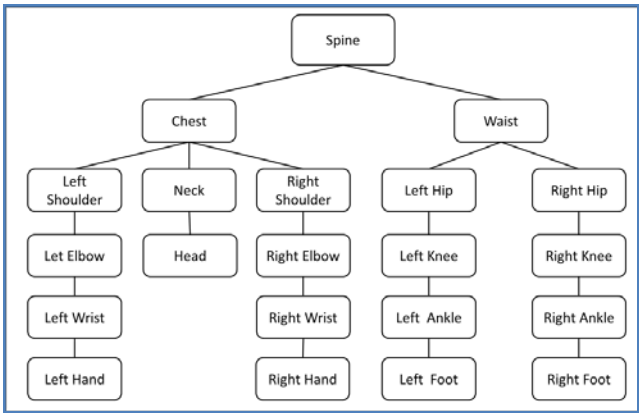

Fig. 2. Resulting stickman model

3.1. Preprocessing

Preprocessing stage consists of two steps, which is foreground detection and image clipping as shown in Fig. 3. Foreground detection producing a background removed depth image and image clipping producing clipped area to minimize computation on the next stage.

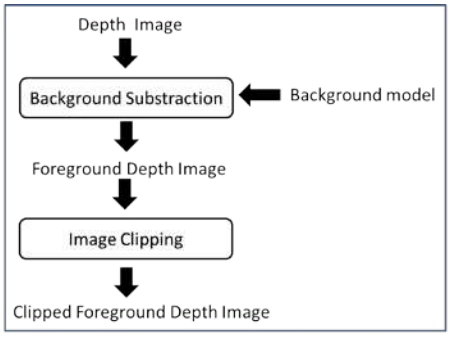

Fig. 3. Preprocess stage flow

Foreground detection is a method to separate foreground with the background from a given full image. (Hamissi & Faez, 2013) implementation do the foreground segmentation by finding the minimum object range on an image, and keep the nearest object as foreground. On (MA, XU, & WANG, 2013) implementation, a foreground image of hand body part is extracted with the help of Microsoft Kinect SDK. Palm information is extracted from the SDK and is used as a reference depth of the hand. On our solution, foreground detection is done by comparing input image with a background image model pixel by pixel. If a pixel has a difference more than determined threshold, the pixel can be considered as a foreground part, if not, the pixel can be removed. On our solution, we use a static background model to optimize the speed of the whole process. Example of foreground detection is shown in Fig. 4.

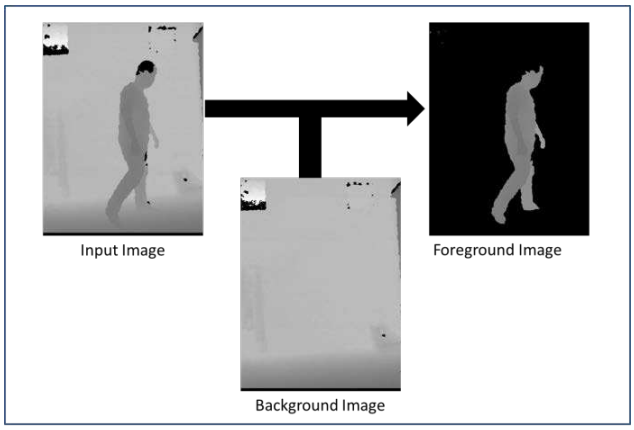

Fig. 4. Foreground detection method

Image clipping is a method to take only some part of full image. By taking only a partial image, the next stage calculation could be optimized. On our solution, image clipping is done by using a rectangular clip. The border of the clip is determined from the foreground depth image border. Example of image clipping is shown in Fig. 5.

Fig. 5. Image clipping result

3.2. Haar Cascade Face Detection

Face detection is a method to detect any faces from the input image by using integral image. Haar cascade is one popular method to do face detection task. Haar cascade detect face by using Haar-like features. There are four types of Haar-

like feature, which is edge feature, line feature, center-surround feature, and special diagonal line feature. These features are improved with 45˚ tilted feature from the original feature. Illustration of Haar-like features is shown in Fig. 6. The extracted feature from each figure is a value obtained by subtracting sum of pixels under the white area with sum of pixels under the black area.

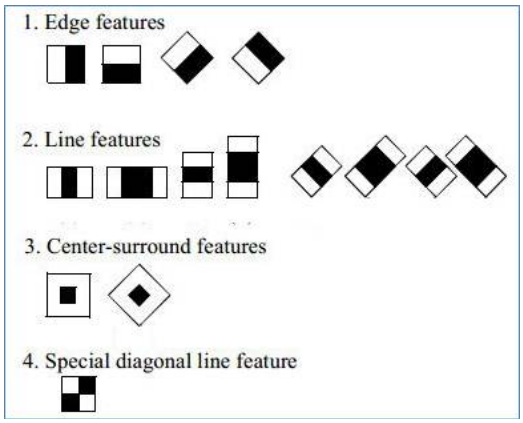

Fig. 6. Haar-like features

The features then processed with machine learning algorithm, usually by using AdaBoost. The training model created from AdaBoost than use for classify input image. Classification is done by using a scalable window. Then feature is extracted from the window. If the extracted features match with the training model, then we can classify the window is a face region. The result from the face detection classification is collection of windows containing the face region. Fig. 7 shows result of the Haar cascade face detection.

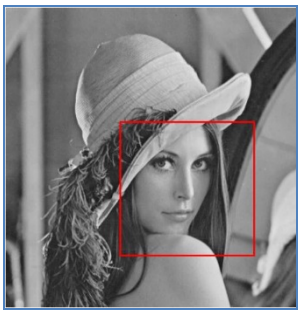

Fig. 7. Haar cascade face detection result

3.3. Distance Transform

Distance transform is commonly used in computer vision and pattern recognition. Our distance transform use chessboard distance. Chessboard distance assigns distance value of current pixel by counting minimum range from current pixel to the nearest background pixel. Illustration of chessboard distance is shown in Fig. 8.

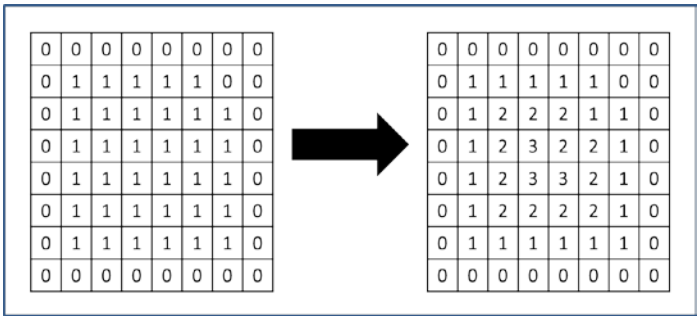

Fig. 8. Chessboard distance example

After distance transform process is performed, we perform pixel deletion with a simple rule by checking 8 neighboring pixels from every pixel that is not background. If the current checked pixel has neighbor pixel with distance value more than itself, the current checked pixel will be deleted. After deletion, the remaining pixel is local maximum pixel that located on the center of each body part. Our distance transform result is shown in Fig. 9.

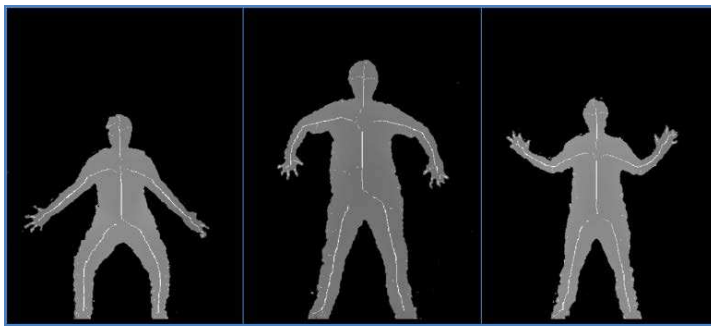

Fig. 9. Distance transform result

3.4. Anthropometry Analysis

We use anthropometric data to get general percentage comparison of human body. The anthropometric data is summarized from NASA Anthropometric Source Book Volume 1: Anthropometry for Designers [5]. We split the percentage comparison into 3 human body part, which is core body, arm, and leg. The core body part is divided into 6 part, which is head, neck, neck to bust, bust to stomach, stomach to waist, and waist to hip. The arm part is divided into upper arm, lower arm, and hand. The leg part is divided into thigh, calf, and foot. The percentage of core body, arm, and leg part is shown on Table 1. Illustration of each part is shown in Fig 10.

Table 1. Body Part Percentage

Body Part	Percentage
Core BodyPart	
Head	25.19589894
Neck	7.942145734
Neck to Bust	16.61662395
Bust to Stomach	19.86085683
Stomach to Waist	19.37385573
Waist to Hip	11.01061882
Arm Part	
Upper Arm	44.38514091
Lower Arm	31.44748079
Hand	24.16737831
Leg Part	
Thigh	44.38557163
Calf	40.20786631
Foot	15.40656205

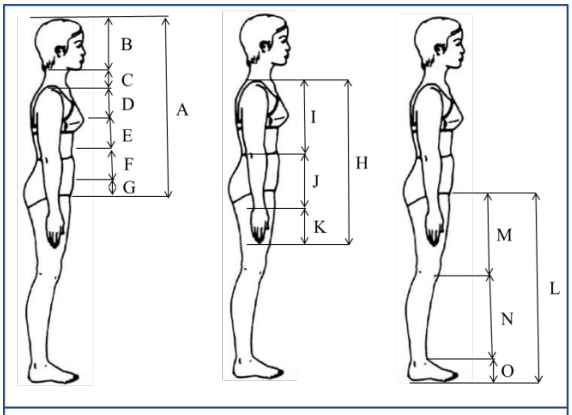

Fig. 10. Body measure

From the anthropometric analysis, we will develop the stickman model. Our stickman model consists of 21 joints from head to foot. The stickman model shown in Fig 10.With the anthropometry percentage and face detection result, we determine the position of all core body part. After that we determine the joint location of head, neck, chest, spine, center hip, left hip, right hip, left shoulder, and right shoulder.

In order to determine the thigh, calf, and foot joints, we trace path from left and right hip to foot and create a prefix sum list of path range. After that we calculate the range of knee joint and ankle joint by calculating anthropometric percentage of leg part with the total path range and determine which pixel become the candidate of the knee, ankle, and foot joint. We use the same method to calculate the arm joints but is start from left and right shoulder joint.

.

IV. Experimental Result

Our experiment is done by using Intel Core I5 2.67 GHz processor with 8 GB DDR3 RAM. The prototype is implemented in C# by using Microsoft Kinect SDK to retrieve RGB and depth Image and AForge.NET library. We test our method solution with 480x640 image resolution. The average running time of our solution shown in Table 2. Comparison of our solution with (Jain & Subramanian, 2011), (Kar, 2010), and the Microsoft Kinect SDK is shown in Table 3

Table 2. Running time per stage

Stage	Avg. Time	Min Time	Max Time
Preprocess	~16.971 ms	8 ms	78 ms
Face Detection	~58.198 ms	20 ms	221 ms
Distance Transform	~108.375 ms	67 ms	180 ms
Anthropometric Analysis	~0.125 ms	0 ms	8 ms
Total	**~183.669 ms**	**95 ms**	**478 ms**

Table 3. Comparison with other implementations

Aspect	Our Solution	(Kar, 2010)	(Jain & Subramanian, 2011)
Detected Joint	21 joints	8 joints	8 joints
Avg. Time	~183.6 ms	103 ms	68 ms
Detected body part	Whole body	Upper body	Upper body
Image Resolution	480x640	640x480	320x240

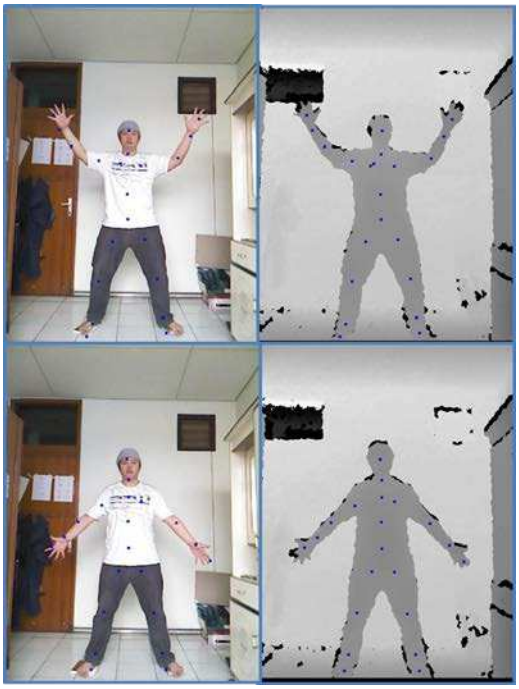

Fig. 11. Estimated joints overlay on RGB and depth image

REFERENCES

[1] Dedeoglu, Y. (2004). *Moving Object Detection, Tracking and Classification for Smart Video Surveillance.* Ankara, Turkey: bilkent university.

[2] Gritai, A., & Shah, M. (2006). Tracking of human body joints using anthropometry . *Int. Conf. of Multimedia and Expo.* Florida: University of Central Florida.

[3] Hamissi, M., & Faez, K. (2013). Real-Time Hand Gesture Recognition Based on the Depth Map. *International Journal of Electrical and Computer Engineering*, 2-4.

[4] Jain, H. P., & Subramanian, A. (2011). Real-time Upper-body Human Pose Estimation using a Depth Camera. *HP Laboratories*.

[5] Kar, A. (2010). *Skeletal Tracking using Microsoft Kinect.* Kanpur: Methodology.

[6] MA, B., XU, W., & WANG, S. (2013). A Robot Control System Based on Gesture Recognition. *TELKOMNIKA, Vol. 1, No. 5*.

[7] NASA. (1978). *Anthropometric Source Book Volume I: Anthropometry for Designers.* Springfield: NASA.

[8] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., et al. (2011). Real-Time Human Pose Recognition in Parts from Single Depth Images. *IEEE Conference.* Microsoft Research Cambridge & Xbox Incubation.