3D HAND POSTURE RECOGNITION USING MULTICAM

MOHD NAZRIN BIN MUHAMMAD
NURDIANA BINTI NORDIN

(2011 IEEE International Conference on Signal and Image Processing Applications (ICSIPA 2011),
16-18 November 2011, Hotel Maya, Kuala Lumpur)

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

3D Hand Posture Recognition using MultiCam

Mohd Nazrin bin Muhammad*[1] and Nurdiana binti Nordin#[2]
*Faculty of Manufacturing Engineering, Universiti Teknikal Malaysia Melaka, Melaka, Malaysia.
# Faculty of Electrical Engineering, Universiti Teknikal Malaysia Melaka, Melaka, Malaysia.
[1]Email: nazrin@utem.edu.my, [2]Email: nurdiana@utem.edu.my

Abstract— **This paper presents the hand posture recognition in 3D using the MultiCam, a monocular 2D/3D camera developed by Center of Sensorsystems (ZESS). The MultiCam is a camera which is capable to provide high resolution of color data acquired from CMOS sensors and low resolution of distance (or range) data calculated based on time-of-flight (ToF) technology using Photonic Mixer Device (PMD) sensors. The availability of the distance data allows the hand posture to be recognized in z-axis direction without complex computational algorithms which also enables the program to work in real-time processing as well as eliminates the background effectively. The hand posture recognition will employ a simple but robust algorithm by checking the number of fingers detected around virtually created circle centered at the Center of Mass (CoM) of the hand and therefore classifies the class associated with a particular hand posture. At the end of this paper, the technique that uses intersection between the circle and fingers as the method to classify the hand posture which entails the MultiCam capability is proposed. This technique is able to solve the problem of orientation, size and distance invariants by utilizing the distance data.**

Keywords—**MultiCam, hand posture, hand posture recognition, real-time processing, 3D, distance data**

I. Introduction

Hand posture is defined as a constant hand image presentation without regard to location, orientation and time in space but represent to specific class such as palm, fist, expanded five fingers etc. However, hand postures exist in variants of size and orientation. The different color adds to the complication of the recognition. On a brighter side, hand posture contains limited configurations. Yet, different people will not be able to represent similar configurations since they tend to show the configurations in their own ways.

Hand posture recognition has gained public attention as an alternative to Human-Computer Interface (HCI) and the methods of the hand posture recognition could be divided into the way how the hand features are extracted. Since the high-level is exclusively for the 3D hand model-based, other feature extraction techniques lie on either image-based of low-level features.

In image-based features, a few works have been used such as Principal Component Analysis (PCA) which put a variation of hand postures into the eigenspace [14][15]. The aim of using PCA is to reduce the dimensionality of the features by extracting the most significant features only. Another method in image-based features that have been applied is the Haar-like features [12]. The objective of the Haar-like features is to reduce in-class variability and at the same time increase the out-of-class variability, thus making the classification easier and faster.

Other method of image-based features includes replacing original RGB images into LUV color space [4]. Then, a scale-space representation is computed for each LUV channel by convolution with Gaussian kernels. Blob analysis is applied on the scale-space representation to find the finger regions. Figure 1 shows the result of the works [13][4]:

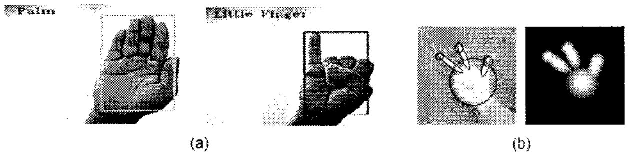

Figure 1: (a) Posture recognition based on Haar-like features: on the left shows "Palm" is detected and on the right shows the "Little Finger" is detected [13]. (b) Selected image features resulted from blob analysis is shown on the left and the mixture of Gaussian kernels is on the right [4].

Another solution for the feature extraction is to extract low-level features in which the geometry and moments from the hand posture images are analyzed. The assumption lies on the fact that the hand is anatomically the same for every person and therefore some features that belong to the hand itself such as the fingers and the shape are utilized.

In an example, the fingers of the hand are in nature protruded out of the palm and by making a circle centered at the hand's centroid, the profile of the circle which intersect with the fingers are easily detected as shown in Figure 2(a). This has been done in [11] and [8]. Besides that, the fingers and palm could create some shape profiles when lines are projected onto them in vertical and horizontal axes [16] as shown in Figure 2(b). The same class of hand postures are said to have a similar profiles and the classifiers could use them to discriminate among other classes.

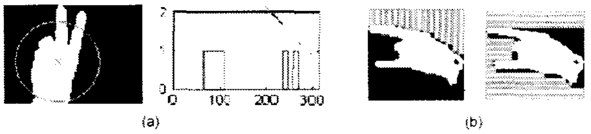

Figure 2: (a) A circle is superimposed into hand image on the right and the intersections are shown in zero-to-one transition diagram on the left. (b) The vertical and horizontal projections are shown on the left and right figures respectively [16].

Although a number of research works have been proposed, most of them are restricted to certain orientation and hand's coordination. For example, the accuracy of the projection technique is very sensitive to any orientation different from the training's orientation. It is also susceptible to the size and the distance of the hand during the training process. These problems happen to other techniques too. In order to have a good robustness of the hand posture recognition, the solution to be chosen should be less affected with any of the hand's orientation, size and distance in respect to the camera.

II. MULTICAM AS COLOR AND RANGE DATA PROVIDER

At the Center of Sensorsystems (ZESS) in University of Siegen, a new 2D/3D imaging system has been developed which could extract color and distance information. It has been employed in an integrated system between the imaging system and program known as *Hactor*, that capable to detect the present of hand in a complex background [5].

The 2D/3D imaging technique is referring to a monocular hybrid vision system, named MultiCam (see Figure 3), which is able to provide color and range data. The camera is embedded with a Photonic Mixer Device (PMD) chip which measures the range data (depth information) using Time-of-Flight (ToF) principle [7][10].

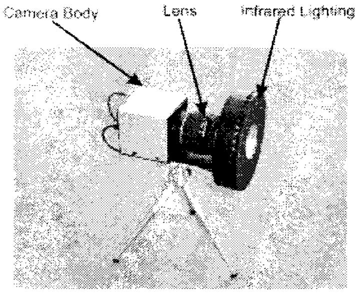

Figure 3: The MultiCam [5].

The component of the MultiCam consists of several parts. The PMD sensor which is realized from the CMOS technology has pixels window of 64x48 that receives range and modulation data. To acquire the high resolution 2D image (640x480 pixels), the camera has integrated with the CMOS sensor.

In order to measure the distance, ToF can be measured by measuring the phase delay between the transmitted and received light signal to extract the distance data. This method is also known as continuously-modulated (CW) or sinusoidally modulated. The CW method is much preferred to be used because it requires lower requirement for electronic components.

In order to calculate the distance between camera and each point of the scene, the autocorrelation function of electrical and optical signals is analyzed by a phase-shift algorithm [17]. Using four samples A1, A2, A3 and A4 with each shifted by 90 degrees, the phase which is proportional to distance, can be calculated using the following equation:

$$\varphi = arctan\left(\frac{A_1 - A_3}{A_2 - A_4}\right) \qquad (1)$$

In addition to the phase-shift, two other values can be extracted. At first, the signal strength of the received signal (amplitude) is calculated as

$$a = \frac{\sqrt{(A_1 - A_3)^2 + (A_2 - A_4)^2}}{2} \qquad (2)$$

The second value which represents the gray-scale value of each pixel, offset b, is calculated as

$$b = (A\_1 + A\_2 + A\_3 + A\_4)/4 \qquad (3)$$

Lastly, the distance d is calculated as follows:

$$d = (c.\varphi)/(4 \cdot \pi \cdot f\_mod) \qquad (4)$$

where fmod = modulation frequency and c= speed of light.

III. HAND POSTURE RECOGNITION

Hand posture refers to the position or orientation of the hand without regards to the time. Thus the hand posture recognition is a process[2] (see Figure 5) to group certain hand posture images into some predefined classes where each class is said to have a common essential property or group of properties.

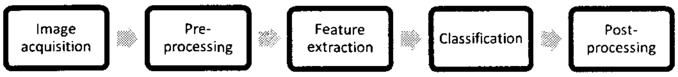

Figure 5: Hand posture recognition process

Researchers have previously worked on many articles [12][14][16] to provide solutions to the hand posture recognition which comprise of two approaches: 3D hand model-based and appearance-based. Although 3D hand model could provide the highest number of recognized hand postures, high accuracy is almost impossible to attain unless the users have to wear additional equipment such as glove or markers, which makes the human-computer interactions unnatural.

Another option is the appearance-based which is the best option to do the hand posture recognition because it could extract hand features directly from the images and computationally efficient. Furthermore, in this work, the input data of the hand posture is received as an image which makes the appearance-based approach as the best option to choose.

A. Types of posture

Each hand posture can be represented into many meaningful definitions for example when the index finger points to upwards direction while the others contracted could be referred as number 1 or pointing up or a symbolic way to say 'listen' or 'warning'.

Another example is shown in Figure 6, where the hands showed the same finger's configuration but each of them has different definition.

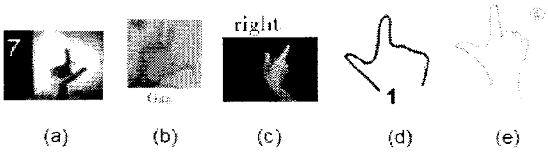

Figure 6: Different definitions from the similar hand's orientation: (a) represents the number '7' [3] (b) represents a symbol of 'gun' (c) to show

the direction to the right and (d) & (e) represents the Class 1[9] and 4[1] of hand posture respectively.

The variation of meanings is influenced by cultures, environment, education, and others. Thus, it is important to have a good idea of how to set the meanings of each of the hand posture images so that their definitions are explicit and relevant to use in accordance to the gesture classifications.

B. *Posture recognition with MultiCam*

In order to attain robustness of the hand posture recognition, the proposed solution which uses the intersection between the circle and fingers as the method to classify the hand [8] offers recognition ability invariant to hand's orientation, size and distance with respect to the camera.

This technique is able to produce a good result due to several reasons. First, it exploits the natural shape of the hand anatomy which has a center of mass at the center of the hand palm. When a circle's center is drawn at the center of the mass of the hand and with a suitable radius of the circle, all the fingers will be intersected with the circle as fingers are protruded along the perimeter of the palm.

The intersections between the circle and the fingers will provide the knowledge of how many fingers are extended at that time. A few classes can be set for the different hand postures. Also, due to its nature, this technique is not affected by the orientation of the hand as the orientation only makes the fingers to rotate around its center of mass, within the perimeter of the circle.

Moreover, this technique is appropriate because the size of the circle to be drawn is dynamically adjusted based on the distance. With the range information, the size of the circle can be controlled so that it will change dynamically relative to the hand's size.

The range information also improves the quality of the hand image projected into the screen significantly. This helps the algorithm to achieve better result in determining the intersection of the circle and the fingers. The intersection (binary transition) for found finger and no-found finger along the circle perimeter, is important as a way to determine the hand posture's classes and a noisy image can produce unreliable and inconsistent result of the classes.

C. *Methodology*

The chosen technique works by virtually placing a circle at the center of the mass (CoM) of the hand image. The center of the mass of the hand image (as shown in Figure 7) is formulated as follows:

$$x_{center\ of\ mass} = \frac{\sum_{1}^{n} x_{hand\ image\ pixel}}{n} \qquad (5)$$

$$y_{center\ of\ mass} = \frac{\sum_{1}^{n} y_{hand\ image\ pixel}}{n} \qquad (6)$$

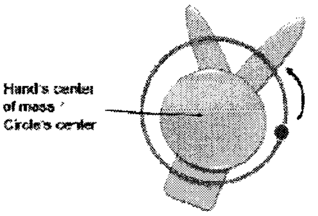

Figure 7: The visualization of center of mass of the hand and virtually created circle

The size of the circle to be placed on the hand is determined by two factors; the average size of the human/operator hand and the distance between the hand and the camera. Since this work will be used and tested by adults, samples of adult hands have been taken and the average sizes of hand's parameters are calculated.

TABLE II
SAMPLES OF ADULT HAND POSTURES

	Finger Width		Finger Tip to Center of Mass		Finger Length	
	close	far	close	far	close	far
Person 1	36	21	255	187	171	125
Person 2	32	18	258	185	164	115
Person 3	26	21	235	163	149	105
Person 4	24	15	223	153	141	96
Average	30	19	243	172	156	110

CLOSE: 120-125CM, FAR: 185-190CM. ALL OTHER VALUES ARE IN PIXELS. THE MENTIONED FINGER REFERS TO THE MIDDLE FINGER.

Based on the sizes information, the radius (rad) of the circle to be constructed can be determined by taking the average length from the center of mass (CoM) to the half length of the finger.

$$\overline{rad}\left(\frac{close}{far}\right) = \overline{finger\ tip\ to\ CoM} - \frac{\overline{finger\ length}}{2} \qquad (7)$$

Through this, the circle is ensured to intersect with the middle finger within the closest limit (close) and the farthest limit (far).

In order to vary the radius of the circle, range information provided by the MultiCam is analyzed. This is to change the radius into smaller value when the distance between the camera and the hand becomes farther. There are two steps to be taken: first is to calibrate the provided range value with respect to the real range value and secondly to calibrate the size of the circle according to the closest limit and the farthest limit.

D. *Range and Circle Calibration*

Upon experimenting with the MultiCam, it is found that the range value provided by the camera is not reflecting the actual distance. Therefore, a test has been done to record the differences between the actual and the measured ranges. The test was done by recording the value measured by camera with different exposure time over some predetermined distance. The objective of this test is to obtain an equation that will convert measured distance into actual distance. The following table shows the actual and measured data collected from the test.

TABLE III
RESULTS FROM RANGE CALIBRATION TEST

		Exact distance	122	131	144	151	159	170	181	189
Exposure time	5ms	Measured distance	154	155	165	172	181	192	201	208
	10ms		152	155	164	176	184	192	201	212
	15ms		152	153	168	176	184	192	206	215
Average			153	154	166	175	183	192	203	212

(ALL VALUES ARE IN CM)

After the range calibration, the calibrated range data can be used to dynamically resize the radius of the circle. The idea is to make the circle intersects at all the fingers even if the distance of the hand and the camera is changed. The circle must be resized appropriately so that the size is big enough when the hand is at the closest limit to the camera and small enough when the hand is at the farthest limit from the camera. The closest limit is in this case is 120 cm and the farthest limit is 190 cm where the closer or the farther limits respectively will cause the hand image disappears from the screen. This range is also known as Volume of Interest (VoI), set originally from the *Hactor* program.

In regards to this idea, there is a simple solution to set the initial circle's radius as if the hand is at the closest limit which is 140 pixels as per discussed before. To make the radius becomes smaller with the farther distance; it is multiplied with a ratio of closest limit over current calibrated distance. The equation is as follows:

$$adj.radius, r_{adj} = 140 * \frac{closest\ limit}{current\ calibrated\ distance} \quad (8)$$

Thus, if the current calibrated distance equals to closest limit, the circle retains its initial preset value and if the current calibrated distance becomes larger (farther away from the camera), the radius will become smaller. Although this technique could work properly, but there is no mechanism to control the radius size at the farthest limit. Therefore, in order to accommodate the control mechanism for easier future changes, the equation is added with a constant multiplier and an offset as follows:

$$adj.radius, r_{adj} =$$
$$140 * \frac{120}{current\ calibrated\ distance * multiplier} + offset \quad (9)$$

To obtain the values of the two unknowns, the multiplier and the offset, a minimum of two set of equations are needed to be equate. The steps to calculate the unknowns are shown as follows:

At closest limit:

$$r_{adj} = 140 = 140 * \frac{120}{120 * multiplier} + offset \quad (10)$$

At farthest limit:

$$r_{adj} = 110 = 140 * \frac{120}{190 * multiplier} + offset \quad (11)$$

E. Binary Transition

After the necessary distance and circle calibrations, the next task is to acquire the binary transition of the finger and the circle intersection. The transition is a series of binary of 0 and 1 represents the no-finger and finger detection respectively. The series is also known as transition vector. The vector normally contains a fixed number of elements which is determined by a step size (in degree) as the transition is count around the circle's perimeter. To illustrate this, the next figures show how the transition vector is obtained.

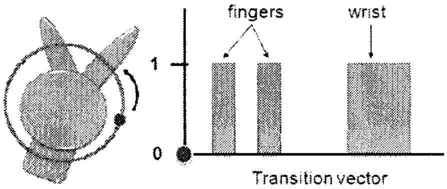

Figure 8: A virtual circle and its corresponding transition vector.

The first step to form the transition vector is to set a fix initial point on the circle as indicates by the gray ball. Next, from the initial point either from clockwise or anticlockwise (in this example is anticlockwise), the point is moved along the circle (visualize in red color). The point will move with a step size of one degree and to complete the path, a total of 360 data will be available as the transition vector. For each data, it represents either 0 for the background color detected (or non-skin color) and 1 for the non-background color detected (or the skin color).

As shown in Figure 8, there are a number of 1s recorded representing the two fingers and the wrist. A finger is detected when the size (width) of the 1s lies between the lower and upper limits that have been set for a finger. As the width of the finger varies with the distance of the hand and the camera, the set limits must be change dynamically. The method to determine the dynamic equation for the limits is similar to the circle calibration.

F. Finger's Width Calibration

Suppose that there are 1s in the transition vector and there is a group of 1s adjacent to each other that will be label as W_i to indicate that the width of the group might represent a finger. Both lower and upper limits change from initial values at the closest limit and decrease at the farthest limit.

$$if\ W_{low}(lower\ limit) < W_i < W_{upp}(upper\ limit)$$

then a finger is detected, else W_i is not a finger.

Lower limit calibration

At the closest limit:

$$W_{low} = 15 = 15 * \frac{120}{120 * multiplier} + offset$$

At the farthest limit:

$$W_{low} = 5 = 15 * \frac{120}{190 * multiplier} + offset$$

Hence:

$$W_{low} = 15 * \frac{120}{d_{cal} * 0.55} - 12 \qquad (12)$$

Upper limit calibration

At the closest limit:

$$W_{upp} = 50 = 50 * \frac{120}{120 * multiplier} + offset$$

At the farthest limit:

$$W_{upp} = 28 = 50 * \frac{120}{190 * multiplier} + offset$$

Hence:

$$W_{upp} = 50 * \frac{120}{d_{cal} * 0.84} - 9.7 \qquad (13)$$

G. *Image Pre-Processing*

Image quality plays a big role in determining a good result for this technique. The current input image is taken from the original *Hactor* program and the image contains hand image with a gray background. The image poses a good quality image with almost no noise in the background with the help from the range information of the MultiCam. The image is constructed by taking the color details from the original image which has the same depth as the hand as shown in Figure 9 (a) and (b).

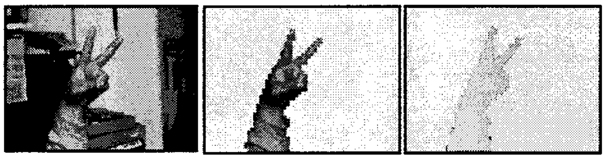

Figure 9: (a-left image) The original image. (b-middle) The segmented hand image. (c-right) The hand image after region growing process.

H. *Transition Vector from Input Image*

Since transition vector is a result of intersection between the hand image and the circle, the object recognition algorithm firstly utilized the formula for the circle to achieve it.

$$kth\ column^2 + kth\ row^2 = {r_{adj}}^2 \qquad (14)$$

To check whether a point (column$_i$, row$_i$) is on the circle or not, the square root of the summation of squared column$_i$ and squared row$_i$ must be very close to the r$_{adj}$.

If a point is in a circle:

$$r = \sqrt{kth\ column^2 + kth\ row^2}\ and\ r = r_{adj} \qquad (15)$$

As the radius of the circle is known, the algorithm could be programmed to continuously check from the first column to the last column in the first row and repeated all over again with the next row until it reaches to the last row of the image. The data for transition vector is then acquired by checking the predefined condition.

To quench more processing time, a simpler method is proposed in which only selected pixels are checked rather than the whole pixels in the image. The method uses rotational matrix to rotate an initial point on the circle which has known location, by a step size of one degree. By using this technique, the total number of pixels that needs to be checked is only 360 pixels and the transition vector is already arranged in the proper arrangement. The algorithm is shown as follows:

Known variables:

1) $center\ of\ the\ circle =$ $center\ of\ mass\ of\ the\ hand = (X_c, Y_c)$
2) $radius\ of\ the\ circle\ based\ on\ the\ range = r_{adj}$
3) $step\ size = 1° = 0.0175\ radian$

Setting the initial point:

$$(X_0, Y_0) = (X_c + r_{adj}, Y_c)$$

Setting the next point:

$$\begin{bmatrix} X_i \\ Y_i \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \end{bmatrix}$$

where $\begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$ is the rotational matrix and θ is the degree in radian.

The simplified version of the algorithm is as follows:

```
for degree from 1 to 360 with step size 1
        $X_i$ = cos (degree)*$X_0$ – sin(degree)*$Y_0$
        $Y_i$ = sin (degree)*$X_0$ + cos(degree)*$Y_0$
        if(color($X_i$, $Y_i$)==GREEN)  //green is the color of
                                          region growing
                transition vector = 1;
        else
                transition vector = 0;
        end if
end for
```

The sample results of the virtual circle and the transition vector are shown in the figure below. The number of finger based on the width of 1s is two which means the hand posture will be recognized as the Class 2.

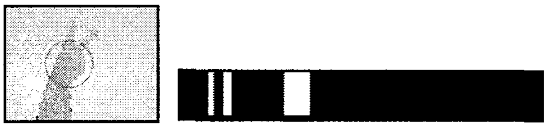

Figure 10: Left image is the hand image with the circle is drawn centered at the CoM. Right image is the resultant of the transition vector plotted in a bar-graph style.

IV. RESULTS AND DISCUSSION

In the hand posture recognition tests, three people are asked to move their hand freely in the space with different hand postures predefined. The tests were done separately for each person. For a given time (i.e. one minute), they are asked to show only one type of hand posture, move front and backward and sometimes try to rotate the hand in z-axis. After the time limit, they are asked to stop and get ready for the next posture. During the test, the results of the binary transition, hand posture class, segmented hand image and original image for every frame are saved into the disk. Table I is the confusion matrix that shows the results from the three tests.

TABLE IV
THE CONFUSION MATRIX FOR HAND POSTURE RECOGNITION

Test Posture (as shown by the user)	Result (as shown by the program)					Performance
	0	1	2	3	4	
0	711	9	1	0	0	98.61%
1	44	632	12	1	0	91.73%
2	41	8	615	6	0	91.79%
3	42	6	29	668	0	89.66%
4	48	0	12	34	736	88.67%

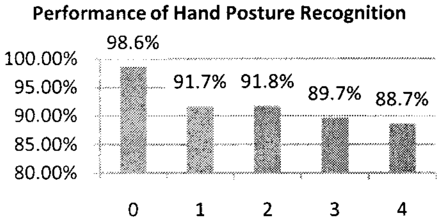

Figure 13: The bar chart of the performance of the hand posture recognition of the program.

In reference to Figure 13, the performance of the hand posture recognition of the program is shown. In the first sight, hand posture of Class 0 has the best performance and followed by the Class 1 and 2. Class 3 and Class 4 showed performance below 90% with Class 4 is the worst.

V. CONCLUSION

The hand posture algorithm manages to identify and recognize five different types of hand posture with average performance around 90% mainly due to the poor hand image quality. However, the range information helps to improve the quality of the images and enables the algorithm to dynamically change its parameters to suit the recognition process.

The failure rate of hand posture recognition can be further reduced by finding the solution to improve the hand image quality and by increasing the number of training images respectively. The performance of hand posture recognition can be improved if a higher resolution of range data is made possible as better quality hand image can be produced.

The output obtained from the hand posture recognitions could be extended to gesture classifications and other works such as machine control or software control applications. As a whole, the depth information provided by the MultiCam enables more complex applications feasible in the future especially in the image processing.

VI. ACKNOWLEDGEMENTS

This work has been supervised by Dr.-Ing. Klaus Hartmann, M.Sc. Seyed E. Ghobadi and Dipl.-Phys. Omar E. Loepprich from ZESS, Siegen and funded by UTeM and the Government of Malaysia in the form of scholarship for the postgraduate study.

REFERENCES

[1] Shin JH, Lee JS, Kil SK, Shen DF, Ryu JG, Lee EH, Min HK, Hong SH. Hand Region Extraction and Gesture Recognition using Entropy Analysis. Int. Journal of Computer Science and Network Security. 2006 216-222.

[2] MarquesdeSa, J.P., Pattern Recognition: Concepts, Methods and Applications. Heidelberg: Springer-Verlag Berlin Heidelberg New York, 2001

[3] Nielsen ES, Canalis LA, Tejera MH. Hand Gesture Recognition for Human-Machine Interaction. Journal of WSCG, Vol. 23, No. 1-3. 2003.

[4] Bretzner L, Laptev I, Lindenberg T. Hand Gesture Recognition using Multi-Scale Color Features, Hierarchical Models and Particle Filtering. Proc. of the 5th IEEE Int. Conf. on Automatic Face and Gesture Recognition.

[5] Ghobadi SE, Loepprich OE, Ahmadov F, Bernshausen J, Hartmann K, Loffeld O. Real Time Hand Based Robot Control Using 2D/3D Images. ISVC 2008. 2008 307-316.

[6] Yang MH, Ahuja N, Tabb M. Extraction of 2D Motion Trajectories and Its Application to Hand Gesture Recogntion. IEEE Transactions on Pattern Analysis and Machine Intelligence. 2002 1061-1074.

[7] Ghobadi SE, Hartmann K, Weihs W, Netramai C, Loffeld O, Roth H. Detection and Classification of Moving Objects - Stereo or Time-of-Flight Images.

[8] Malima A, Ozgur E, Cetin M. A Fast Algorithm for Vision-Based Hand Gesture Recognition for Robot Control. Signal Processing and Communications Applications. 2006:1-4.

[9] Liu Y, Liu X, Jia Y. Hand-Gesture Based Text Input for Wearable Computers.

[10]Beder,C., Bartczak,B., Koch,R. A Comparison of PMD-Cameras and Stereo Vision for the Task of Surface Reconstruction using Patchlets., Minneapolis, 2007.

[11]Yin X, Xie M. Hand gesture segmentation, recognition and application. Computational Intelligence in Robotics and Automation, 2001. Proceedings 2001 IEEE International Symposium on. 2001:438-443

[12]Fang Y, Cheng J, Wang J, Wang K, Liu J, Lu H. Hand posture recognition with co-training. 19th International Conference on Pattern Recognition. 2008:1-4.

[13]Chen Q, Georganas ND, Petriu EM. Hand Gesture Recognition Using Haar-Like Features and a Stochastic Context-Free Grammar. Instrumentation and Measurement, IEEE Transactions on. 2008:1562-1571.

[14] Akmeliawati R, Dadgostar F, Demidenko S, Gamage N, Kuang YC, Messom C, Ooi M, Sarrafzadeh A, SenGupta G. Towards real-time sign language analysis via markerless gesture tracking. Instrumentation and Measurement Technology Conference. 2009:1200-1204.

[15]Wu,Y., Huang T.S, View-independent Recognition of Hand Postures, Proc. IEEE Conference on Computer Vision and Pattern Recognition, vol 2, 2000, pp.88-94.

[16] Deng JW, Tsui HT. A Novel Two-Layer PCA/MDA Scheme for Hand Posture Recognition.; 2002; Quebec City.

[17] Ringbeck T. A 3D Time of Flight Camera for Object Recognition Detection. [Internet]. 2007 [cited 2010 Jan 5]. Available from: http://www.pmdtec.com/fileadmin/pmdtec/downloads/publications/200705_PMD_ETHZuerich.pdf.