HAND GESTURE RECOGNITION, TRACKING, AND PROJECT TO A 3D MODEL

[1] *Dayo Choul* (*邱大祐*), [2] *Zheyuan Gao* (*高哲远*) , [3] *Shiten Huang* (*黃士展*), *Chiou-Shann Fuh* (*傅楸善*)

Department of Computer Science and Information Engineering,
National Taiwan University, Taipei, Taiwan 106
[1] E-mail: b98902037@ntu.edu.tw
[2] E-mail: gaozheyuan13@gmail.com
[3] E-mail: b98902101@ntu.edu.tw
[4] E-mail: fuh@csie.ntu.edu.tw

ABSTRACT

In this paper, we introduce a low-complexity and low-cost method to recognize human gestures. Instead of using machine learning method, we design a motion capturing and 3D modeling framework, consisting of existing techniques and concepts, which are connected strongly with each other to precisely recognize hand gestures and project to a 3D model. First, we apply motion detection to capture the moving objects in the sequences of frame, and predict the possible movements or gestures. Synchronously, the hand color will be estimated to apply calibration. So far the possible hand regions and basic movements are detected, in order to recognize more complex gestures and track the moving path precisely, feature detection is next. Not only hand but also finger points are detected by overriding some Haar features. Finally, after distance are also estimated, we can construct stereo information and interact with a 3D model.

Keywords *Hand gesture recognition; Hand tracking; Project from plane to 3D model; Computer human interaction*

1. INTRODUCTION

Hand gesture recognition has been a very common way in computer human interaction. Most devices use stereo camera system, e.g. Kinect, Leap Motion, and so on. To reduce the cost and complexity, we use a simple webcam to develop a real-time hand gesture recoginition and 3D projection model. Due to the USB (Universal Serial Bus) bandwidth constraints, we cannot control multiple webcams simultaneously, our task is not only to build a low-cost and low-complexity system, but also to recognize common gesture and tracking precisely. In short, we propose a low-complexity but reliable method effectively to meet user's basic needs by using just one monocular webcam. Althogh monocular camera has its restriction, e.g. depth information, this technique is still effective for common users. For instance, the Samsung Galaxy S4 is a smartphone manufactured by Samsung Electronics. It was announced in New York City on March 14, 2013. The phone has many distinguishing software features such as Air Gesture, which has applied gesture recognition by just a front monocular camera. The front camera has 2 mega pixels and 30 fps [9].

Most methods without stereo camera use glove-based method, which means this interface requires the user to wear a cumbersome device, and carry many cables that connect the device to a computer, apparently user-unfriendly. Another method is vision-based method also model-based method [2].

David and Shah [3] propose a model-based approach by using a finite state machine to model four distinct phases of a generic gesture. Hand shapes are described by a list of vectors, and these vectors are used to match vector models. Starner et al. [8] use binary camera, build a system to track hands in real time, and interpret sign language. They use Hidden Markov Models (HMMs) to recognize a complicated series of gesture. Cui and Weng [10] propose a non-HMM-based system that can recognize 28 different gestures under complex backgrounds. The recognition accuracy is 93.1% but segmentation phase takes 58.3 sec. for each image not recommended for a real-time system. Zhu and Yuille [6] develop a statistical framework using principal component analysis and stochastic shape grammars to represent and recognize the shapes of moving objects, which is also called Flexible-Object Recognition and

Modeling system (FORMS). Lockton et al. [5] develop a real-time gesture recognition system that can recognize 46 English alphabets ([a-z, A-Z]) and digits ([0-9]). The recognized gestures are static and still.

Different from the method mentioned above, we propose a simple and reliable system to to recognize dynamic gesture in real-time, in complex backgrounds and infer 3D position to build a human-computer interface. Our system has four sub-systems: basic image processing, calibration and prediction, real-time hand tracking and recognition, and projection.

Basic image processing consists of four basic image processing steps: motion detection, skin color extraction, edge detection, bounding box, labeling.

Different from previous model-based gesture recognition system, we introduce a Calibration and Motion-based Prediction Framework (CMPF) to replace model-based recognition system. It is a motion-based system, tracks the moving hands, analyzes the hand shape, and plans the search space while. Moreover, it does background substraction more precisely and cooperates with the other systems simultaneously. CMPF continues feedbacking results to the system in order to maintain the performance and accuracy. By integrating CMPF and basic image processing, ROI (Region Of Interest) and basic motion variation can be detected approximately. To precisely recognize more complex gestures, we propose CMPF, in our experiment, to test some existing methods and find tracking and prediction useful for recognition. Because most gesture recognition methods enter the recognition phase at last step, which means user might feel recognition delay accompanied with bad User eXperience (UX). In order to optimize UX, we apply CMPF to create an illusion that recognizing phase is accelerated.

In feature extraction sub-system, we modularize Haar-like features [4], by using template-matching method, the feature points. For example, fingers, phalanges, and metacarpal can be detected in this system. In our survey, Wang Yinghui et al. [11] propose a contour-based method by calculating the curvature of contour. We find it too complicated due to large quanity of calculation and system performance deterioration. It is precise but difficult to maintain in a real-time system. When feature points are detected, we add each points to a vector for the next phase. We have experimented with several feature detection methods, e.g. SIFT [7], SURF… and find that feature detection costs huge calculations without pre-model training. In order to accelerate feature detection without model-based method, we use GPU resource by embedding Nvidia CUDA (Compute Unified Device Architecture) SDK (Software Development Kit).

In 3D projection system, we integrate the previous works, using area size information to infer 3D position, and calculate the variance of feature vectors between each frame. In front-end layer, we build a simple 3D interface interacting with users.

Motion tracking and gesture recognition are handled separately in our system.

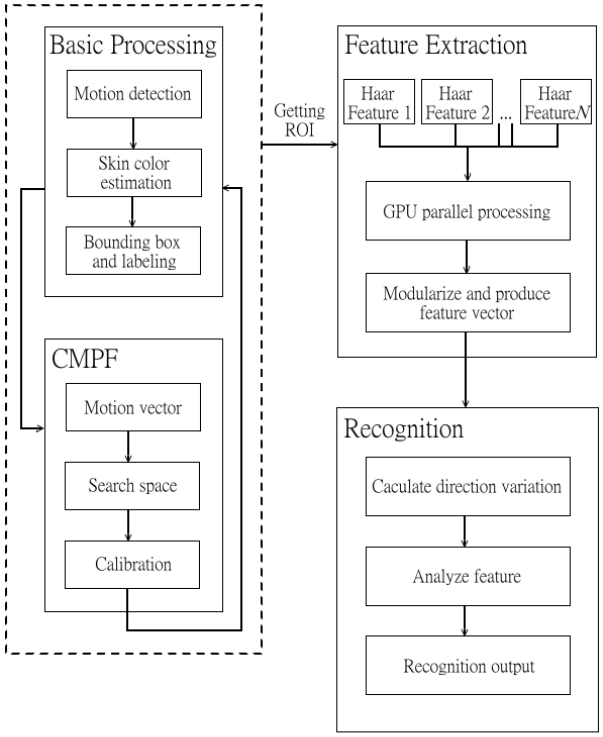


Fig. 1. The flow chart of hand tracking and gesture recognition system

When hand is moving, we only track the motion. In tracking phase, we focus on calculating motion direction variation and calibration. When hand or gesture is still, the system starts recognizing gesture. In recognition phase, feature points in hand region will be described as a vector, then matched with the exsiting vectors. Scale and rotation will be handled as well in this phase. Before vector matching, the existing vector set is stored no matter it is one of vector set in our system of gesture or any user-customized gesture.

2. BASIC IMAGE PROCESSING

In this part, we implement some basic image processes, and those works are used for the next phase, CMPF. Due to large computations in next phases, we need to take trade-off problem in to account, which means, in this phase, low complexity, simpilicity, and robustness is the primary task that we concern instead of accuracy. In this phase, we provide the basic functions to extract hand and moving region roughly. We need to deal with the

explosure time which can not be controlled, unless we edit the driver of our webcam. Thus when hand moves rapidly, some outliers might be counted unavoidably.

2.1. Motion Detection

Motion detection simply examines the gray-level changes in each frame. As long as the target object is moving, it can be detected. A hypothesis says that environment light source should be stable. Otherwise, it will have an impact on experiment result and lower the reliability significantly. As mentioned before, the explosure time is also a variable of our system, and rapid movements might cause wrong bounding box which should be handled in CMPF. Let $F_i(x, y)$ be the i-th frame in an image sequence; $D_i(x, y)$ be the absolute difference between i-th and $(i+1)$-th frame [10]; and x and y are the coordinates in the 2D space of frame. The function is defined as below [10].

$$D_i(x, y) = |F_i(x, y) - F_{i+1}(x, y)|$$

However, calculating the difference between each frame is not enough for obtaining rough hand region, because of the variation caused by the environment light source and low resolution of webcam along with much noise in $D_i(x, y)$. In order to denoise, we should set a threshold for $D_i(x, y)$. The denoising function is defined as follows [10].

$$D_i(x, y) = 1, \text{ if } D_i(x, y) > Threshold.$$
$$D_i(x, y) = 0, \text{ otherwise.}$$

Yet when applying thresholding for denoising, threshold value is not determined depending on hardware. Besides, Otsu's method is not suitable for moving object capturing, because it will generate too much noise [10]. Fig. 2 shows the result of $D_i(x, y)$ with our threshold setting. Bouding box generates a restriction region, preparing for the following steps and phases.

Fig. 2. $D_i(x, y)$ with the threshold value 80 and bounding box.

2.2. Skin Color Estimation

Skin region is detected by using color estimation. We use HSV color space rather than RGB color space. The image in RGB was converted to HSV color space, because it is more related to human color perception [1] and more intuitive for ranging. The skin in channel H is constrained in value ranging from 0 to 50, in the channel S from 0.23 to 0.68 for Asians and Caucasians [7].

Fig. 3. The skin color estimation marks the skin color in red.

However, in Fig. 3 we find that the skin color will include a wide range of hand color like regions [10], e.g. face, neck, and other non-human objects. With the motion detection in Section 2.1, we have bounding-box moving regions, so we use the information produced by motion detection to remove redundancy and obtain "real moving hand", the result is shown in Fig. 4.

Fig. 4. The redundant information in skin color estimation image is removed by using the result of motion detection.

We also find that in dark environment, lack of enough light source, if threshold value is unchanged, we might lose some information of moving hand shape. In order to

overcome this problem and make sure the result in this phase will not be affected by environment light source, we apply mathematical morphology to enhance the shape of the moving hand region. Fig. 5 shows the mathematical morphology enhancement.

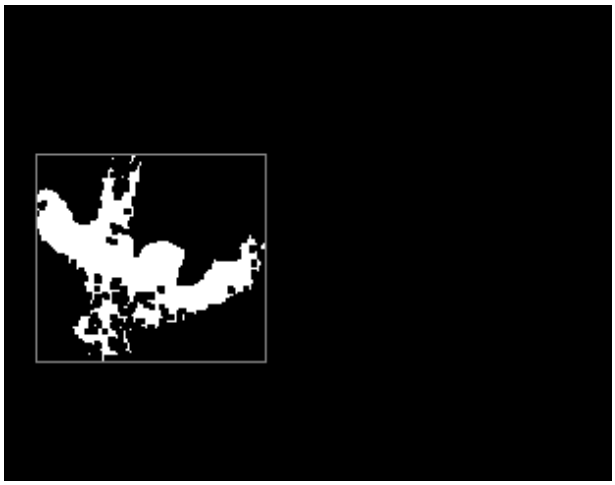


Fig. 5. Note that this image is taken in the dark environment. After mathematical morphology, the hand shape is stronger compared with its original.

3. CALIBRATION AND MOTION-BASED PREDICTION FRAMEWORK (CMPF)

If we put recognition and tracking in the last phase, user might feel the recognition delay and bad UX, which are not appropriate for a real-time system, either. To improve and optimize UX, we propose Calibration and Motion-based Prediction Framework (CMPF), including motion vector, search space, and calibration, to make UX more fluent. It runs only in background and feeds back its outcome to other sub-system in real time.

3.1. Motion Vector

Since we already have moving hand region, which is local information in the previous phase, to reduce the calculations, we use correlation to do image matching locally, instead of matching in the whole image. We create a 15×15 block to match between $F_i(x, y)$ and $F_{i+1}(x, y)$; find the minimized x-y offset for each block. The minimized matching function is below:

$$\sum_{(x,y)\in S} | PIX(F_i(x,y)) - PIX(F_{i+1}(x+dx, y+dy)) |$$

where S denotes the region which is a combination of moving hand region and search space in Section 3.2; $PIX(F_i(x, y))$ denotes the pixel value of image F on coordinate x and y. Similarly, $PIX(F_{i+1}(x+dx, y+dy))$ denotes the pixel value of image F_{i+1} on coordinate $x+dx$ and $y+dy$ where dx and dy are the offset of block that should be minimized in each iteration to find the maximum likelihood in local neighborhood. According to the calculated dx and dy, the information of motion variation for each block we have assigned is obtained, so the moving direction is determined now. In Fig. 6, a visible motion vector of two continuos images is shown.

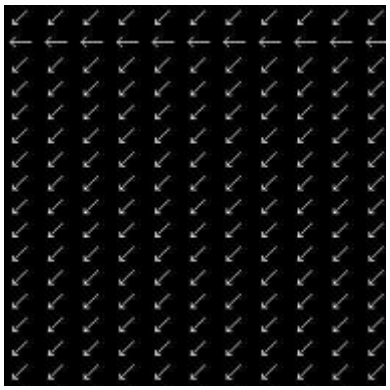
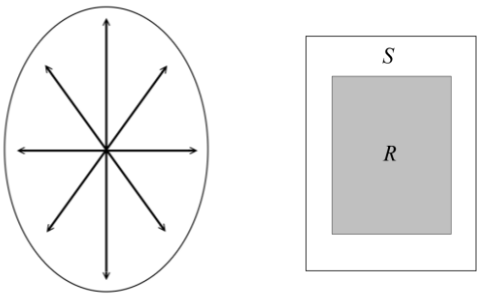


Fig. 6. An example of visible motion vector

3.2. Search Space



(a) (b)

Fig. 7. (a) A sketh map of search space. (b) A group of search space and hand region

In Fig. 8, search space is composed of neighborhood relationship. It not only assists motion vector but also projects all candidates of its possible movement next time. First, when calculating motion vector locally, the computation area is determined by grouping hand region and its search space candidates. We find that if we only calculate hand region, the outcome is not precise without importing search space, because the result of motion vector of object and background is stronger than object itself. The center in Fig. 7(a) denotes the centre of hand region bounding box, and its radial lines denote the space searching direction. Fig. 7(b) denotes the actual condition of grouping search space and hand region.

Second, search space can be used to predict next movement of the target object and reduce the tracking time. With the prediction, we do not need to do basic image processing and motion vector in each frame; thus we do not track object in each iteration. Once when there is any variation in search space, we can simply regard it as direction moves. Althogh this method might

cause error sometimes, it still has the advantage of low complexity and intuition.

3.3. Calibration

Although integrating motion vector and search space is almost sufficient for object tracking calibration, there is still another problem to deal with. Actually, the system has to consider the real practice of common users, for instance, when user waves palm to scroll the visible part of screen up or down, there is high probability that palm moving back to its original calibration location rapidly. Unnecessary reverse direction judgement will occur if they want to scroll up or down continuously.

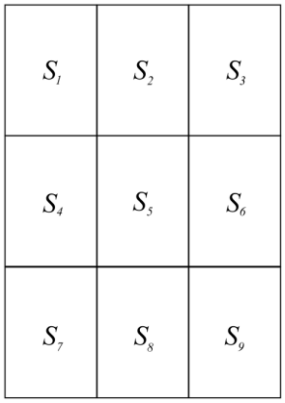


Fig. 8. Search space S is separated into 9 equivalent blocks to help observe regular pattern of motion.

We observe that there is a regular pattern in this kind of movements. In Fig. 8, the palm starts waving at S_5, going to S_2, and back to S_5… The system might detect up, down, up, down… but user just wants to scroll up continuously. To overcome this problem, when any direction movement is detected, we only feed back to that particular direction moving in effective time T_e; T_e can be extended if motion is still detected. In T_e, except for that particular movement, the other will be suspended. We show an NFA (Nondeterministic Finite Automaton) in Fig. 9.

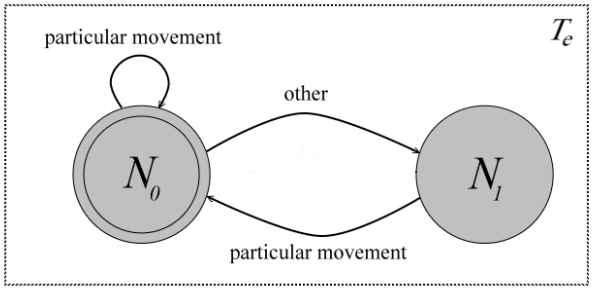
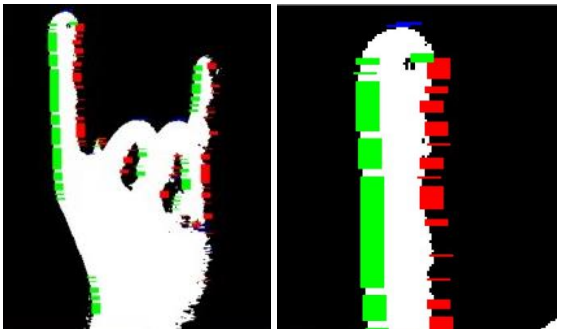


Fig. 9. Current motion will trigger T_e. Only one particular movement will be accepted by calibration system in finite T_e.

4. FEATURE EXTRACTION

Haar feature is widely used in object recognition, e.g. facial features, and we use it to detect hand features such as finger, phalanges, and metacarpal. Our system starts to extract feature only when hand is still. In implementation, we do template matching for each Haar-like features parallelly on each frame because template matching is trivial. If we do it linearly the system is suspended until all features have been searched in local area where the motion is detected last time. In Fig. 10, we show a visible image of Haar-like feature template matching. Different colors denote different Haar-like features.



(a) (b)

Fig. 10. (a) An example of template matching. (b) A large scope of Fig. 10 (a). Three different features are found around finger.

In Fig. 1, our flow chart shows that we do Haar-like feature template matching parallelly by using GPU resources. After matching all features, we combine some of them to form a real feature, e.g. finger. For instance, in Fig. 10(b), we can see three features around a finger, we can use the spatial relationship to modularize a finger model. Fig. 11 shows the result of modularization, and some finger points are clearly seen.

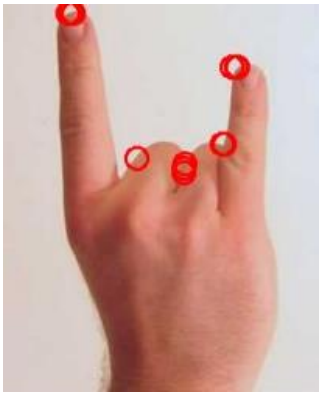


Fig. 11. Haar-like features form feature points after modularization and are labeled with red circles.

Since we have extracted feature points, we can describe as a vector A_f in two-dimensional space by setting the relative coordinates of feature points and center as true

in bounding box. In projection, we use the size, width and height of bounding box to infer z-axis.

5. RECOGNITION

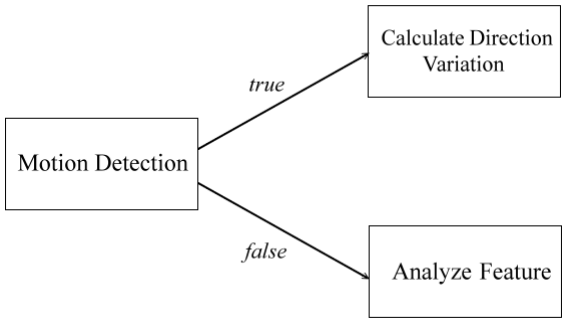


Fig. 12. A flow chart of recognition phase in our system.

Basically, our system separates hand movement into two kinds. Fig. 12 shows the flow chart of recognization. If motion is detected, our system regards it as that user wants to control direction. In Section 3, we handle the motion variation and feed the direction information back to front-end. If no motion is detected, object is still, then we trace back to the area where motion was detected just a moment ago, calibrate, and start to analyze the feature vector vector A_f in Section 4.

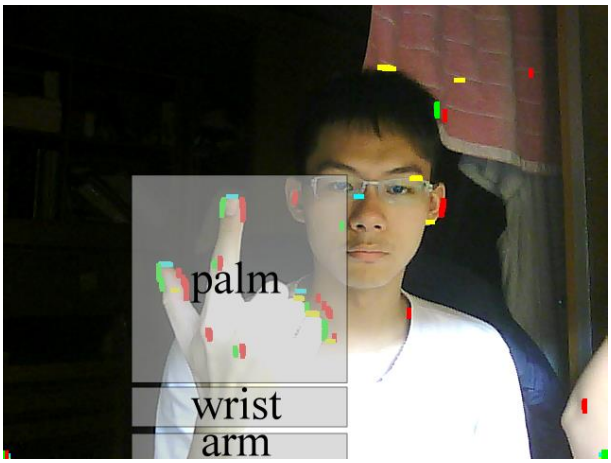
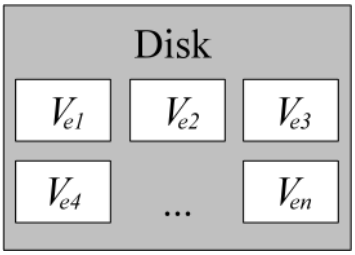


Fig. 13. Hand gesture is composed of palm, wrist, arm in the order top-down.

The scale and rotation might affect the analysis. To overcome this problem, we deal with rotation first. Commonly, for a user sitting in front of a laptop or PC, when he gestures, the hand structure must be palm, wrist, and arm in the top-down order in an image. We show an example image in Fig. 13. Based on this hypothesis, we can determine right or left hand, and label thumb, forefinger… to pinkie. After labeling each finger of user's palm, we align A_f with default gesture vector A_D in the order of labels, (A_D denotes the default vector of a hand gesture with all fingers stretched and palm toward to webcam) and scale A_f to the same size as A_D.

Since alignment is done, we calculate the angle of A_f and each A_e which denotes the existing feature vector set or user-customized gestures which have been analyzed and stored in our system. Fig. 14 shows the pseudo code of our recognition method.




```
Rotate(Vf);
Scale(Vf);

for( i = 1 to n){
        θ = arcos(Vf • Vei / |Vf||Vei|);
        if ( θ ≒ 0)
                break; /*matched with Vei */
}
```

Fig. 14. Pseudo code of our recognition method.

6. CONCLUSION

Althogh we have surveyed some model-based methods, e.g. HMM (Hidden Morkov Model), and we know there are many existing hand gesture datebases. It is trivial to build one. Instead of using model-based method, our objective is to build an open-source library and let developers use their own databases and apply our C++ (CPP) classes free. We focus on extensibility to simplify the recognition phase. While the other parts are complete, we provide the basic but absolutely enough information in our system to make recognition phase independent with the others. No matter developers want to use it to train their own models, build an expansion version or our system, or even develop an application, we believe the extensibility of our system is convenient for them.

ACKNOWLEDGEMENT

This research was supported by the National Science Council of Taiwan, R.O.C., under Grants NSC 98-2221-E-002 -150 -MY3 and NSC 101-2221-E-002 -194, by Winstar Technology, Test Research, and Lite-on.

REFERENCES

[1] A. Albiol, L. Torres, and E. J. Delp. "Optimum Color Spaces for Skin Detection," *Proceedings of International Conference on Image Processing*, Thessaloniki, Greece, Vol. 1, pp. 122-124, 2001.

[2] F. S. Chen, C. M. Fu, and C. L. Huang, "Hand Gesture Recognition Using a Real-Time Tracking Method and Hidden Markov Models," *Image and Vision Computing*, Vol. 21, pp. 745–758, 2003.

[3] J. Davis and M. Shah, "Visual Gesture Recognition," *IEE Proceedings -Vision Image Signal Processing*, Vol. 141, No. 2, pp. 101-106, 1994.

[4] P. Viola and M. jones, "Rapid Object Detection Using a Boosted Cascade of Simple Features," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Providence, RI, Vol. 1, pp. 511-518, 2001.

[5] R. Lockton and A. W. Fitzgibbon, "Real-time Gesture Recognition Using Deterministic Boosting," *Proceedings of British Machine Vision Conference*, Cardiff, UK, pp. 1-10, 2002.

[6] S. C. Zhu and A.L. Yuille, "FORMS: A Flexible Object Recognition and Modelling System," *Proceedings of International Conference on Computer Vision*, Cambridge, Massachusetts, pp. 465–472, 1995.

[7] S. L. Phung, A. Bouzerdoum, and D. Chai, "Skin Segmentation Using Color Pixel Classification: Analysis and Comparison," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, No. 1, pp. 148-154, 2005.

[8] T. Starner and A. Pentland, "Visual Recognition of American Sign Language Using Hidden Markov Models," *Proceedings of International Workshop on Automatic Face-and Gesture-Recognition*, Zurich, Switzerland, pp. 1-6, 1995.

[9] Wikipedia, "Samsung Galaxy S4," http://en.wikipedia.org/wiki/Samsung_Galaxy_S4, 2013.

[10] Y. Cui and J. J. Weng, "Hand Sign Recognition From Intensity Image Sequences with Complex Backgrounds," *Proceedings of IEEE International Conference on Automatic Face and Gesture Recognition*, Killington, Vermont, pp. 259-264, 1996.

[11] Y. H. Wang, W. Y. Wu, and R. J. Zhao, "Segmentation and Recognition Techniques for Planar Contour," *Journal of Computer-Aided Design and Computer Graphics*, Vol. 14, No. 12, pp. 1142-1151, 2002.