Illumination-invariant hand gesture recognition

América Ivone Mendoza-Morales[a*], Daniel Miramontes-Jaramillo[a**] and Vitaly Kober[a,b***]

[a]Department of Computer Science, CICESE, Ensenada, B.C. 22860, Mexico
[b]Department of Mathematics, Chelyabinsk State University, Russian Federation

ABSTRACT

In recent years, human-computer interaction (HCI) has received a lot of interest in industry and science because it provides new ways to interact with modern devices through voice, body, and facial/hand gestures. The application range of the HCI is from easy control of home appliances to entertainment. Hand gesture recognition is a particularly interesting problem because the shape and movement of hands usually are complex and flexible to be able to codify many different signs. In this work we propose a three step algorithm: first, detection of hands in the current frame is carried out; second, hand tracking across the video sequence is performed; finally, robust recognition of gestures across subsequent frames is made. Recognition rate highly depends on non-uniform illumination of the scene and occlusion of hands. In order to overcome these issues we use two Microsoft Kinect devices utilizing combined information from RGB and infrared sensors. The algorithm performance is tested in terms of recognition rate and processing time.

Keywords: Hand gesture recognition, Kinect, occlusion, non-uniform illumination.

INTRODUCTION

Due to advancements in technology, new human-computer interaction methods such as hand gesture recognition, body gesture recognition, and voice command recognition have been developed and improved in recent years. In this work, we focus on hand gesture recognition because it is an interesting problem for device control, mouse substitution, augmented reality and virtual reality interaction, among many other applications.

Hand gesture recognition is composed of two basic steps: identify the hand(s) in each frame of a video sequence, and recognize the user's gesture from a set of available hand gestures.

The vision-based approach is widely used for hand gesture recognition because it provides independence between the user and devices. The input data is captured with device cameras such as smartphones, tablets and intelligent TVs using integrated cameras as well as some laptops and PCs.

Recently, introduction of affordable depth cameras into market has increased the interest of vision applications in industry and research, because it provides novel ways to solve problems exploiting a new kind of information such as the depth information.

However, even with the use of improved cameras and algorithms, many problems in the hand gesture recognition remain unsolved, such as illumination invariance and occlusion, among others. A classic approach to provide illumination invariance is to convert the RGB input of a camera into different color spaces such as YCbCr.[1] We utilize new available information provided by depth cameras along with RGB integrated cameras.[2] The occlusion problem is approached through the use of multiple cameras.[3]

Further author information:
* A.I.M.M. (correspondence): e-mail: amendoza@cicese.edu.mx
** D.M.J.: e-mail: dmiramon@cicese.edu.mx
*** V. K.: e-mail: vkober@cicese.mx

In this work, the stages for hand gesture recognition are as follows: capture the frames using two Microsoft Kinect sensors and detect hand(s) in every frame using the algorithm proposed by Viola-Jones;[4] then the recognition is performed with the help of a support vector machine.

The paper is organized as follows: in section 2, the proposed algorithm is presented; in section 3, computer results are presented and discussed; finally section 4, summarizes our conclusions.

PROPOSED ALGORITHM

The proposed algorithm has four steps: data acquisition, detection, data fusion, recognition. The description of every step is presented below.

Data acquisition

The algorithm input data is captured by depth sensors from two Kinect cameras with a resolution of 640×480 pixels and a frequency of 30 frames per second (fps). Kinect depth information ranges from $800\ mm$ to $4000\ mm$ in normal mode acquisition or from $400\ mm$ to $3000\ mm$ in near mode. In our case we use near mode with a maximum range of $2000\ mm$ to discard irrelevant information. Let $S(x,y)$ be a depth image from the sensor quantized to $8\ bit$ signal after applying a median filter to remove depth sensor noise.[5]

Detection

We use the Viola Jones algorithm[4] to detect a general position of the hand in every frame. The algorithm consists in detecting an object of interest using a classifier based on values of simple features of the target object. Figure 1 shows the features that describe the object;[4] we use the AdaBoost[6] learning algorithm to select relevant features from the object; in our case the target object are hands.

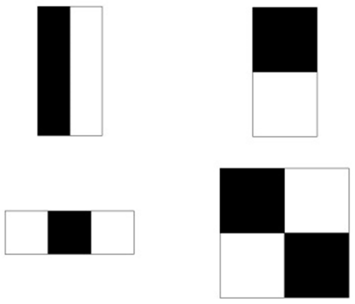

Figure 1. Feature operators.

In order to compute the features, we utilize an integral image.[7] The integral image (SI) is calculated as a sum of pixel values from the left and above to the current position as follows:

$$SI(x,y) = S(x,y) + SI(x-1,y) + SI(x,y-1) - SI(x-1,y-1). \qquad (1)$$

With the integral image we can calculate the sum of pixels in a given region using values of area corners as follows:

$$REG(\alpha) = SI(A) + SI(D) - SI(B) - SI(C), \qquad (2)$$

where $REG(\alpha)$ is a region from which we want to calculate the sum of the pixel values; A, B, C, D are the corners of such region with coordinates (x,y), as shown in Figure 2.

The AdaBoost algorithm constructs a strong classifier $h(x)$ from weak classifiers $h_i(x)$. The weak classifiers are calculated as follows:

$$h_i(x) = \begin{cases} 1, & if\ \ p_i f_i(x) < p_i \theta_i \\ 0, & otherwise \end{cases}, \qquad (3)$$

where $f_i(x)$ is a feature, θ_i is a threshold, and p_i represents the inequality sign.

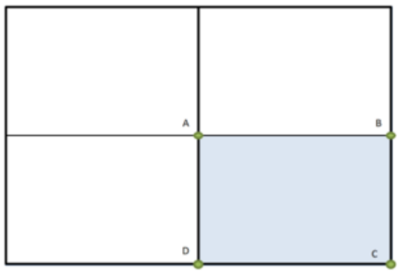

Figure 2. Corners of a region (x, y).

The strong classifier is a lineal combination of weak classifiers, and it is defined as follows:

$$h(x) = \alpha_1 h_1(x) + \alpha_2 h_2(x) + \cdots + \alpha_n h_n(x), \tag{4}$$

where n is the number of features, α_i is the feature rate in the range of $[0,1]$.

Once the target is detected, a region of interest $S_{ROI}(x, y)$ is selected around the detected hand for further processing.

In order to improve hand segmentation, we binarize the region of interest as follows:

$$SB_{ROI}(x, y) = \begin{cases} 1, & S_{ROI}(x, y) \leq Mean(S_{ROI}(x, y)) \\ 0, & otherwise \end{cases}, \tag{5}$$

where $SB_{ROI}(x, y)$ is the binarized frame fragment. Finally, opening and closing morphological operations are applied to reduce residual noise and to obtain a clear hand image.

The next step is to segment the hand in the frame fragment, which is a crucial part for hand gestures recognition. We use convex hull and convexity defects algorithms. Let C be a set of points in the Euclidean plane, the convex hull is the smallest convex set that contains all the points in C.

Then, we calculate the convexity defects given by the convex hull. A convexity defect is a set of points that do not belong to the convex hull.[8] The defect is the space between the hull line and the actual object. We choose the farthest defect between two points in the convex hull for finger characterization.

These algorithms have shown special benefits in hand gesture recognition when applied together for features extraction. The most common features are individual finger segmentation, palm area and hand center, as shown in Figure 3.

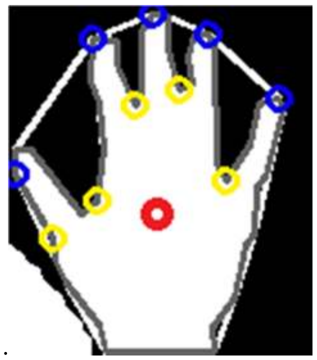

Figure 3. The white line shows the convex hull. The hand contour is marked by gray. The red ring is the hand center, the yellow rings are the convexity defects, and the blue rings are the finger tips.

The features that can be extracted are geometrical or non-geometrical. Some geometrical features are fingertips, finger directions and hand contours. Sometimes these features cannot be calculated owing to light conditions or self-occlusion.[9] In this work geometrical features are extracted using two Kinect cameras. This approach helps us to overcome such common drawbacks.

The features are extracted by means of the convex hull and convexity defects. We also use the hand center of the hand, the distance from the hand center to fingers, the number of fingers, the angles between fingers, the angles between the hand center and fingers, the hand area and hand perimeter.

The hand center, angles and distance from the hand center to fingers are calculated with simple mathematical calculations using the start, depth and end points of convexity defects. Let $CD = \{cd_1, cd_2, \cdots, cd_n\}$ be a set of convexity defects where n is the total number of defects, each element of the set contain three points: start point $s_i(x, y)$, depth point $d_i(x, y)$ and end point $e_i(x, y)$; δ_i is the distance from $d_i(x, y)$ to the edge of the convex hull, $i \in 1, 2, \dots, n$.

The number of fingers nf are computed using the following algorithm:[10]

```
1. Let $minDepth = 20, maxAngle = 60$.
2. For $i = 1$ to $n$
    2.1. If $\delta_i < minDepth$ then continue
    2.2. If $i = 0$ then $predecessor = n - 1$, else $predecessor = i - 1$
    2.3. If $i = n - 1$ then $sucessor = 0$, else $predecessor = +1$
    2.4. Calculate the $angle$ between $s_i(x, y)$ , $s_{predecessor}(x, y)$ and $s_{sucessor}(x, y)$
    2.5. If $angle \geq maxAngle$ then continue
    2.6. $nf = nf + 1$.
```

The hand center $c(x, y)$ is the origin of a minimum circle enclosing the depth points.

The angle between the fingers f_j and f_{j+1} is calculated as

$$\alpha_{f_j} = \tan^{-1}\left|\frac{m_{j+1} - m_j}{1 + m_{j+1}m_j}\right|, \tag{6}$$

where m_{j+1}, m_j are the slope of the line $d_j(x, y)$ to $s_{j+1}(x, y)$, $d_j(x, y)$ to $s_j\ (x, y)$, $j \in\ 1,2, \dots, 5$.

The angle from the hand center to a fingertip j can be computed as

$$\theta_{f_j} = \tan^{-1}|m_j| - 90°, \tag{7}$$

where m_j is the slope to the line from the center to the finger tip $s_i(x, y)$.

Data Fusion

In this stage we fuse the data from both sensors. First, extract the hand features from each sensor image. Then we create a descriptor from the features of both frames, that is, one for each Kinect camera, creating a 26 dimension vector.

Once we compute this vector, we can proceed to gesture recognition.

Recognition

Finally, in this step the hand gesture is classified and recognized. Machine learning algorithms are utilized, and in our case we use a support vector machine (SVM).

The SVM is a supervised learning algorithm commonly used for classification problems. It means if we want to classify a dataset in certain category, first we need to label the dataset, then classify new data in corresponding category.

The training samples are usually mapped to a higher dimensional space. The mapping can be done using kernel functions; which can be lineal, polynomial or exponential. The SVM finds the hyperplane to separate two categories with maximal margin in the higher dimensional space,[11] as shown in Figure 4.

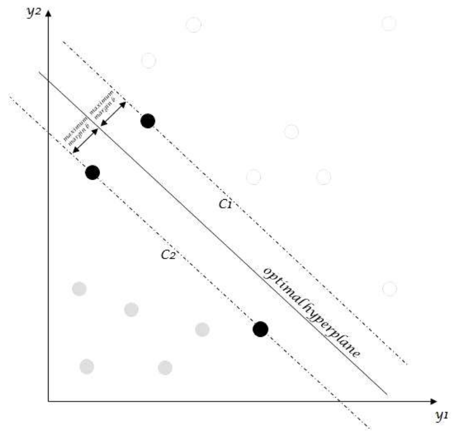

Figure 4. Illustration of linear support vector machine.

The given model can be verified using cross validation, generating an independent data set to test the model.

RESULTS

The proposed system is implemented in a Dell desktop with an Intel(R) Xeon(R) CPU E5-1603, 16GB of RAM, Windows 7 of 64 bits. Images are captured using two Microsoft Kinect for Windows sensors. The implementation of the system was done in C# with the Emgu 2.4.10 wrapper of OpenCV in an OpenSUSE Linux distribution.

To obtain a better view of user's hands and to reduce noise in depth images, one Kinect is located in front of the user and the second one in the left side of the user with an $45°$ angle between the cameras, and a distance of $45\ cm$, the tilt of each Kinect is $16°$. The setup is shown in Figure 5.

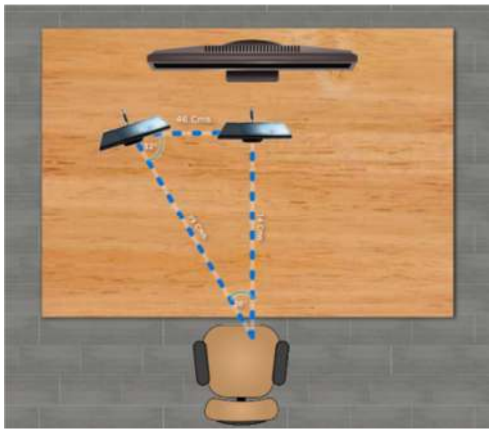

Figure 5. The proposed System setup.

The detection is performed using depth images (640×480 pixels) quantized to $8\ bits$ and preprocessed with a median filter of 13×13 pixels. In order to achieve the hand detection an AdaBoost cascade classifier was trained. 1000 positive sample images and 2000 negative sample images were used. The positive samples were generated from 100 original hand depth images using the software Create Samples.[a] The negative images were taken from our database. The OpenCV Haar training classifier was used to train the system.[b]

[a] http://note.sonots.com/SciSoftware/haartraining.html.
[b] https://github.com/mrnugget/opencv-haar-classifier-training.

Our database contains various depth images. They consist of background and hand images obtained from different distances in the range of $[60\ cm, 200\ cm]$. The hand depth images were taken from 6 people at different distances from $60\ cm$ to $200\ cm$ and for three different poses: open palm and open fingers, open palm and closed fingers, and fist as shown in Figure 6. The negative images were taken from different scenarios as shown in Figure 7. The program to capture the images and to create the database can be found at github.[c]

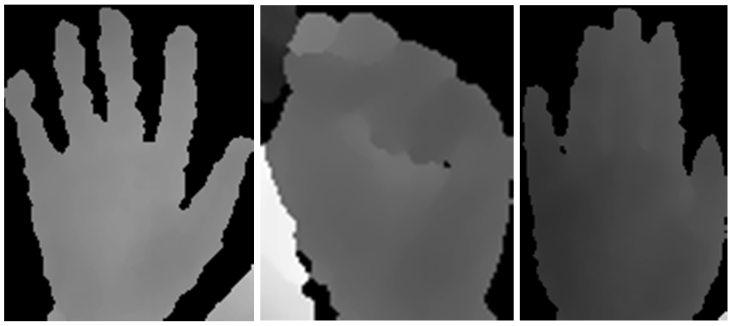

Figure 6. Positive images: left image is open palm; center is a fist; and right image is closed palm.

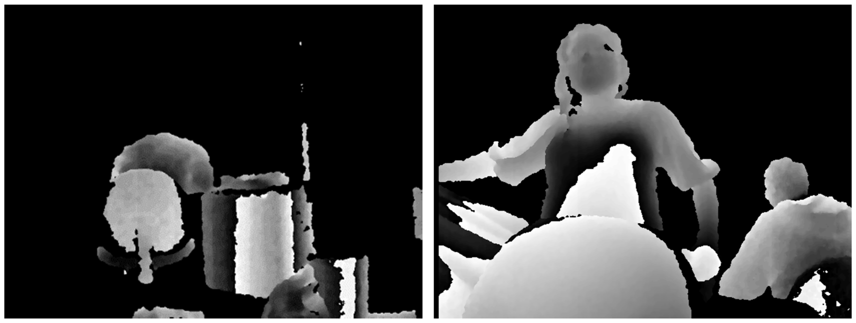

Figure 7. Negative images: left image is office scene and right image is with people in background.

Once a hand is located, segmented and the features extracted, the next step is recognition. We test the algorithm with two static gestures, that is, open palm and fist. A dataset of 120 samples were taken with 60 samples of each gesture; each sample has two views: one with the front Kinect and another with the side Kinect (see Figure 8).

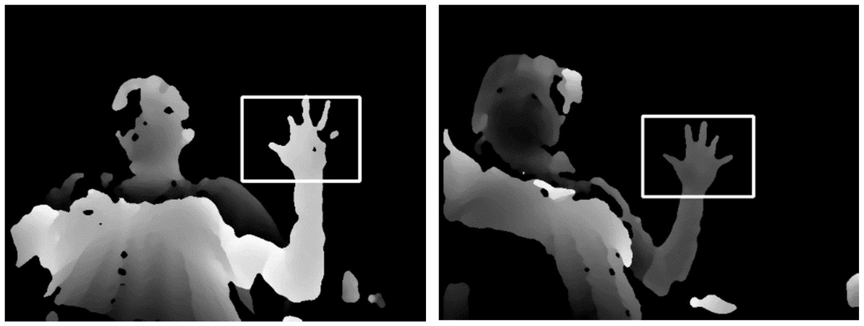

Figure 8. Open palm, right image was taken with the front Kinect; left image was taken with the side Kinect. In both images the hand has been detected.

Two tests were made with the dataset. First, we take 50% of samples to train the SVM and 50% for testing. The second experiment uses 60% of samples to train the SVM and 40% for testing. The training and classification were made using LibSVMsharp[d] (wrapper of LibSVM).[12] For each test an exponential kernel

[c] https://github.com/americamm.

[d] https://github.com/ccerhan/LibSVMsharp.

and cross validations of 5 folds were used. The first test gives a cross validation accuracy of 100% and a test accuracy of 95%. The confusion matrix is show in Table I:

Table I. Confusion matrix with accuracy of 95%.

	Open Palm	Fist
Open palm	27	3
Fist	0	38

The second test gives a cross validation accuracy of 100% and test accuracy of 94.4%. The confusion matrix is shown in Table II:

Table II. Confusion matrix with accuracy of 94.4%.

	Open Palm	Fist
Open palm	32	4
Fist	0	36

CONCLUSIONS

In this work, we proposed a promising algorithm based on only depth information from two Kinect sensors. The algorithm extracts simple, but robust features. We created a hand gesture database and tools to easy segment hands. By using two Kinect sensors we are able to reduce hand recognition errors caused either by object occlusion and self-occlusion. The obtained results showed good accuracy in recognition and classification of gestures. Future work includes improvement of the algorithm to increase the recognition rate using color images and extracting additional features.

ACKNOWLEDGMENTS

The work was supported by Russian Science Foundation grant №15-19-10010.

REFERENCES

[1] Tang, M., “Recognizing Hand Gestures with Microsoft’s Kinect, (2011).

[2] Asaari, M. S. M. and Suandi, S. A., “Hand gesture tracking system using Adaptive Kalman Filter,” *10th International Conference on Intelligent Systems Design and Applications*, 166–171 (2010).

[3] Appenrodt, J., Handrich, S., Al-Hamadi, A. and Michaelis, B., “Multi stereo camera data fusion for fingertip detection in gesture recognition systems,” *Proceedings of the 2010 International Conference of Soft Computing and Pattern Recognition, SoCPaR 2010*, 35–40 (2010).

[4] Viola, P. and Jones, M., “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 (2001).

[5] Mallick, T., Das, P.P. and Majumdar, A. K., “Characterizations of Noise in Kinect Depth Images: A Review,” *IEEE Sensors Journal*, *14*(6), 1731–1740 (2014).

[6] Freund, Y. and Schapire, R., “A desicion-theoretic generalization of on-line learning and an application to boosting,” *Computational Learning Theory*, *55*, 119–139 (1995).

[7] Derpanis, K.G., “Integral image-based representations,” (2010).

[8] Bradsk, G. and Kaehler, A., Learning OpenCV, O’Reilly, Sebastopol, California (2008).

[9] Murthy, G.R.S. and Jadon, R.S., “A Review of Vision Based Hand Gestures Recognition,” *International Journal Of Information Technology and Knowledge*, *2*(2), 405–410 (2009).

[10] Davison, A., “Hand and Finger Detection,” (2013)

[11] Chih-Wei, H., Chih-Chung, C. and Chih-Jen, L., “A Practical Guide to Support Vector Classification,” *BJU International*, *101*(1), 1396–1400 (2008).

[12] Chang, C.C. and Lin, C.J., “Libsvm,” *ACM Transactions on Intelligent Systems and Technology*, *2*(3), 1–27 (2011).