A HOG-BASED HAND GESTURE RECOGNITION SYSTEM ON A MOBILE DEVICE

Lukas Prasuhn, Yuji Oyamada, Yoshihiko Mochizuki, and Hiroshi Ishikawa

Department of Computer Science and Engineering, Waseda University
Okubo 3-4-1, Shinjuku, Tokyo 169-8555, Japan

ABSTRACT

We propose a HOG-based hand gesture recognition system running on a mobile device. Input data is a video of hand gesture taken by a mobile device. The input data is compared with a database storing hand gesture images, which was synthesized with rotation variation. The comparison is done based on their HOG features and the gesture corresponding to the best-matched image is returned as the result. The recognition algorithm is implemented on a client-server system. The proposed system is applied to American Sign Language (ASL) alphabet recognition problem. The experimental results show that the proposed recognition algorithm improves HOG's robustness under rotation change and compare processing time with different network configurations.

Index Terms— Hand gesture recognition, HOG feature, Client-server system

1. INTRODUCTION

Human hand is one of well-used communication tools in our daily life. This visible body action has played an important role in our culture because it enables communication across a language barrier. In particular, speech- and language- impaired people use their hands for daily communication. Recently, hand gesture has been used for human-computer interaction [1].

Using image processing and computer vision techniques for recognizing hand gesture is natural since the target is visual information. One traditional approach for the recognition is the use of data gloves. Data gloves can retrieve accurate hand pose information by using finger bend sensors and have been successfully used for sign language recognition [2]. A recent trend uses depth information for hand gesture recognition [3, 4, 5] inspired by depth image based human pose estimation [6].

Another recent trend in vision community is to develop (existing) vision technique on mobile devices [7, 8, 9]. One of the advantages of mobile vision systems is its portability. Building vision applications on mobile devices, people can receive benefit from the applications in any place where people can go. A key issue in developing such mobile applications is the limited capability of mobile devices, which have several constraints on its computational power, network speed, *etc*.

1.1. Purpose of this paper

In this paper, we propose a hand gesture recognition system running on a mobile device. To compensate the weak processing capability of mobile devices in running accurate recognition while keeping fast processing, the proposed system is built as a client-server system. The user takes a video of hand gesture by a mobile device. Given the picture, the client and the server share the burden of computation over the whole process. Finally, the user receives the recognition result from the server via the client.

This paper has two contributions. One is developing a general hand gesture recognition system on a mobile device. Although we tested the proposed system with a fingerspelling recognition problem, the proposed system was designed to keep its generality. It can be easily applied to other gesture recognition tasks. The other contribution is the robustness of our gesture recognition algorithm. The proposed system adopts the Histogram of Oriented Gradients (HOG) feature. However, HOG is known to be sensitive to object rotation. This disadvantage is remedied by using a database of HOG descriptors corresponding to computer-generated hand images, with which the sensitivity to object rotation can be countered by adding variation in appearances in the database.

2. HAND GESTURE RECOGNITION ON A MOBILE DEVICE

This section describes the details of the proposed hand gesture recognition system on a mobile device. The hand gesture recognition algorithm is described in §2.1

and then the system configuration is described in §2.2.

2.1. Hand gesture recognition algorithm

The proposed system solves the hand gesture recognition problem as a matching problem. The system holds a database that stores a set of hand gesture images, each of which is assigned a label indicating the corresponding gesture. Given a query image, the algorithm compares the query image with the all database images and returns the label of the best-matched image as the recognized gesture. Histogram of Oriented Gradients (HOG) feature [10] is used as the matching metric. The recognition algorithm takes the following three steps: hand area extraction, feature extraction, and feature matching.

2.1.1. Hand area extraction

The recognition algorithm first extracts the hand area in the input image. Hand (or skin) extraction has a long history [11, 12, 13]. Robustness against cluttered situation and, since the hand extraction may be implemented on mobile devices, computational complexity must be considered. The proposed system applies simple binarization in HUV color space. Morphological operations are applied to the binarized image to remove extraction noise. Specifically, the proposed algorithm applies closing first and then opening operations [14].

2.1.2. Feature extraction

Next step is to compute the features of the extracted hand area. The proposed system uses the Histogram of Oriented Gradients (HOG) descriptor [10] to represent hand shape. HOG descriptor describes the shape of the target object by a histogram of intensity gradient. An advantage of the HOG feature is that it is robust under illumination change and shadow; a disadvantage is that its performance suffers under object rotation. Although HOG has its origins in human detection, it is a suitable and popular choice for hand pose estimation [15, 16].

The proposed system follows the parameters used in the original HOG paper [10]. Given an input image of fixed size, HOG gives a feature vector. First, the gradient vector at each pixel is computed. A set of cells, each of which consisting of 9×9 pixels, are defined without any overlap. For each cell, a weighted histogram of gradient orientations is computed. Bins are evenly spread over 0 to 180 degrees, specifically 9 bins with 20 degrees interval are used. Then, a set of blocks, each consisting of 3×3 cells, are defined with overlap. For each block, a feature vector of $9 \times 3 \times 3 = 81$ dimensions is obtained.

2.1.3. Gesture recognition

The final step of the recognition method is to find the best-match image in the database. Each image stored in the database is processed beforehand and has a HOG feature. Any nearest-neighbor algorithm in the feature space can be used. Here, we just use the simplest brute-force matching, since the number of the images in the database remains low. HOG features of the input image is compared with the HOG feature of each database image. The L^2 distance of two HOG features is used as matching metric. The database image with the minimum L^2 distance is selected as the best-matched image. The recognition method returns the assigned label of the best-matched image as the recognition result.

2.1.4. Hand gesture image database

The proposed method uses a database storing a set of hand gesture images. Variation of the database images directly affects the recognition result. The more variation the database has, the more accurate the recognition result becomes. However, with more images the brute force matching takes more time. Since HOG features are robust to illumination change and shadow but not to target object orientation, the database should have variation w.r.t. target object orientation.

Due to the difficulty to correctly label and classify all degrees of freedom of a human hand image, the database images are synthesized by using LibHand [17], which is an open source software that renders a realistic 3D hand model. The model has 21 joint angles which supply 63 degrees of freedom when not constrained by the joint characteristics. Biological constraints are imposed by fixing joints that are not introduced by Lee et al. [18]. Each gesture pose is created by manually adjusting the non-fixed parameters and then a set of images with different orientation is rendered for each gesture pose.

2.2. Client-server system configuration

The proposed system is a client-server system using a mobile device as the client and a desktop PC as the server, those which are connected via the Internet. A user takes a picture of hand gesture by his/her mobile device. The gesture recognition task explained in §2.1 is shared between the client and the server. The user (the mobile device) receives the recognition result from the server.

To obtain the best performance with this client-server system, two issues must be considered. One is computational costs of the recognition method and computational power of the two devices in the system. In general, server

Table 1. Recognition rate of ASL alphabet recognition experiment.

Query \ DB	Frontal view	Multi-view
Frontal view	0.34	**0.47**
Slanted view	0.24	**0.39**

Table 2. The mean and the standard deviation of processing time on each device [msec.].

	Hand extraction		HOG extraction	
	Client	Server	Client	Server
XGA	538.6 ± 67.0	$\mathbf{73.3 \pm 6.2}$	625.0 ± 91.3	$\mathbf{7.3 \pm 1.5}$
SVGA	277.1 ± 42.7	$\mathbf{38.5 \pm 4.6}$	326.4 ± 41.9	$\mathbf{3.2 \pm 1.4}$
VGA	176.9 ± 28.0	$\mathbf{26.5 \pm 4.5}$	209.0 ± 30.8	$\mathbf{2.1 \pm 0.5}$
QVGA	51.2 ± 13.7	$\mathbf{9.9 \pm 2.7}$	64.1 ± 12.7	$\mathbf{1.1 \pm 1.1}$
QCIF	24.5 ± 10.6	$\mathbf{3.8 \pm 1.4}$	34.6 ± 9.8	$\mathbf{0.9 \pm 0.4}$

PCs and mobile devices are unequal in terms of computational power. It is natural that systems assign computationally heavier tasks to server PCs and from this point of view, it is best if the mobile device sends the obtained picture to the server and the server executes all the processes. The other issue is network speed. Since users may bring mobile devices some places where the network speed is slower, client-server systems generally are designed to avoid transferring larger amounts of data. In this sense, it is better to execute HOG feature extraction on the client side and then send the HOG feature, which is smaller data than the original image.

In this paper, two setups are implemented. As described in §2.1.4, the proposed system must hold a hand gesture image database on either the client or the server side. Since it is not desirable to store a large amount of data on mobile devices, we decided to store the database on the server side. This means that the final matching process (§2.1.3) is executed on the server side. We therefore have two choices for the system configuration: (1) the client sends the original picture to the server and the server executes all the processes. (2) the client executes HOG feature extraction and send the extracted feature to the server and the server executes the matching process. In the experiments, we compared these two basic configurations.

3. EXPERIMENTAL RESULTS

For validation of the proposed system, two experiments were conducted: one for recognition accuracy (§3.1) and the other for processing time (§3.2). In both experiments, fingerspelling of American Sign Language (ASL) was used as the target gesture. ASL defines signs for the 26 letters of the English alphabet which are an appropriate candidates for hand configurations with high potential for real world application. These 26 signs are expressed by 19 hand shapes[1].

The HOG descriptor is computed with a total of 64 cells defined by 8 columns and 8 rows. Each cell holds 9 bins that range from 0 to 180 degrees.

For orientation variation, hand gesture of each letter is rendered with 75 uniformly-sampled viewpoints from the interval $[-\pi, \pi]$ for azimuth angle and $[0, \pi]$ for elevation angle. Total number of database images was $1425 = 19 \times 75$.

3.1. Gesture recognition w.r.t. pose variation

We first validated the robustness against pose variation by using two databases. As described in §2.1.4, the proposed system improves HOG's robustness under orientation change by adding rotated images to the database. For this experiment, two databases were created. Frontal view database contains only frontal-view images while Multi-view database also contains rotated variations. The test dataset for this experiment was collected from four persons, who are non-native to sign language. Each person showed 19 signs, each of which has three rotation variations. The three variations are a frontal view and two slanted views, one's azimuth angle is set to either 30° or -30° and the other's elevation angle is set to either 30° or -30°. Note that we don't compare the proposed method with other state-ot-the-art because the recognition algorithm is not the main scope of this paper.

Table 1 shows the recognition rate that is computed by dividing the number of success cases by the total number of queries. In both cases, we obtained better recognition rate with Multi-view database, 47% for frontal view queries and 39% for slanted view queries. Considering state-of-the-art methods [3, 5], 47% of recognition rate is acceptable[2]. Interesting observation here is that the pose variation improves the recognition accuracy even when the query is frontal view. This might be caused by the quality of the query images. Some gesture on the frontal view queries were slightly rotated unintentionally. The recognition algorithm therefore performed better with the Multi-view database than with the Frontal view database.

[1]Each of the groups of letters (H,N,U), (K,P), and (D,G,Q) has a common shape at different rotation and J and Z involve motion.

[2][3] achieves 49.0% and [5] achieves 73.3%.

Table 3. The mean and the standard deviation of processing time through each network.

	LAN [msec.]		SHORT WAN [sec.]		LONG WAN [sec.]	
	RAW	HOG	RAW	HOG	RAW	HOG
XGA	**662.2** ± **88.3**	766.5 ± 93.8	14.3 ± 3.1	**0.8** ± **0.1**	15.3 ± 3.6	**1.8** ± **0.2**
SVGA	**375.5** ± 131.0	377.5 ± **81.6**	10.3 ± 3.2	**0.5** ± **0.1**	12.4 ± 4.8	**1.4** ± **0.1**
VGA	**255.0** ± 45.4	295.4 ± **38.3**	8.5 ± 2.4	**0.4** ± **0.0**	8.7 ± 2.0	**1.4** ± **0.2**
QVGA	**112.7** ± 74.3	118.9 ± **19.6**	5.8 ± 1.8	**0.2** ± **0.0**	7.4 ± 2.8	**1.2** ± **0.2**
QCIF	**92.9** ± 22.1	101.8 ± **21.7**	5.1 ± 1.9	**0.2** ± **0.0**	7.1 ± 2.9	**1.1** ± **0.1**

3.2. Processing time in a client-server system

Two experiments were conducted to evaluate the processing time of the algorithm. We used the following devices throughout the two experiments. A Google Nexus 7 was used for the client device and a desktop PC acted as the server. The client device has a camera with five types of resolution[3], a quad-core 1.5GHz CPU, and 2GB RAM. The server device has an Intel Core i7-2600 processor (3.4 GHz) and 16 GB RAM. Since Nexus 7 uses YUV format for captured videos, hand extraction process involves color convert from YUV to HUV.

3.2.1. Processing time on each device

We measured the time spent for each process on each device to investigate their differences. Table 2 compares the measured processing time. The processing time of HOG feature matching was not considered since the process is always executed on the server side. The ratio of processing time for hand extraction on the client side to the one on the server side was about 5-7 while that for HOG extraction was about 40-100. Considering the real-timeness, 15-30 fps = 33-66 msec., the client device can accept only QCIF while the server can do at most SVGA. The result thus encourages to compute as much process as on the server side.

3.2.2. Processing time through network configurations

The last experiment was conducted to measure the processing time through the entire network that include both computation time and network transfer. We built three different network configurations: LAN, SHORT WAN, and LONG WAN. In LAN setup, the client device and the server device were located in a local network and are connected via a wireless router. SHORT WAN and LONG WAN connected the devices over a 3G mobile network but differ in network path. In LONG WAN network, all packets were routed over a machine physically located far from the server and its processing time took longer than SHORT WAN. We confirmed that the long WAN network just introduced high latency but not significant decrease in bandwidth. We compared these three network configurations with two process sharing setups: RAW and HOG. Input video was sent to the server in the RAW setup while the HOG feature extracted on the client side was sent to the server in the HOG setup.

Table 3 compares all combination of the network configurations and the process sharing setups. Note that the processing time through the LAN network is in the units of msec. while ones through the other networks are in the units of sec. The result shows that the RAW setup depended more heavily on network speed than the HOG setup. The HOG setup is preferrable since mobile devices are used in various networks with different network speed. The established client-server system achieved 0.9-10.0 fps even with QCIF and further improvement in the computation time is therefore encouraged.

[3]XGA (1280×768), SVGA (800×608), VGA (640×320), QVGA (320 × 240), QCIF (176 × 144)

4. CONCLUSION

This paper proposed a hand gesture recognition system running on a mobile device. To support the weak computation power of mobile devices, we built a client-server system. The gesture recognition problem was solved by the system as a matching problem. Given an input image, the system compares the HOG feature of the input image with those of database images and returns the gesture corresponding to the best-matched image.

We applied the proposed system to ASL alphabet recognition and conducted two experiments to evaluate the gesture recognition algorithm and processing speed through networks. The first experiment validated that HOG's sensitivity to orientation is improved by adding rotated images to the database. The second experiment compared the processing time on each device and one through different network configurations. Considering the results, we consider the best solution to be extracting the HOG feature on the client and then executing the latter processes on the server.

References

[1] Victor Adrian Prisacariu and Ian Reid, "Robust 3d hand tracking for human computer interaction," in *Automatic Face Gesture Recognition and Workshops (FG 2011), 2011 IEEE International Conference on*, March 2011, pp. 368–375.

[2] Tomoichi Takahashi and Fumio Kishino, "Hand gesture coding based on experiments using a hand gesture interface device," *SIGCHI Bull.*, vol. 23, no. 2, pp. 67–74, Mar. 1991.

[3] Nicolas Pugeault and Richard Bowden, "Spelling it out: Real-time asl fingerspelling recognition," in *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, 2011, pp. 1114–1119.

[4] Zahoor Zafrulla, Helene Brashear, Thad Starner, Harley Hamilton, and Peter Presti, "American sign language recognition with the kinect," in *Proceedings of the 13th International Conference on Multimodal Interfaces*. 2011, ICMI '11, pp. 279–286, ACM.

[5] Chenyang Zhang, Xiaodong Yang, and Yingli Tian, "Histogram of 3d facets: A characteristic descriptor for hand gesture recognition," in *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*, 2013, pp. 1–8.

[6] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake, "Real-time human pose recognition in parts from single depth images," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Washington, DC, USA, 2011, CVPR '11, pp. 1297–1304, IEEE Computer Society.

[7] Jussi Leppänen and Antti Eronen, "Accelerometer-based activity recognition on a mobile phone using cepstral features and quantized gmms," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2013, pp. 3487–3491.

[8] Phawis Thammasorn, Sukitta Boonchu, and Aram Kawewong, "Real-time method for counting unseen stacked objects in mobile," in *IEEE International Conference on Image Processing (ICIP)*, 2013, pp. 4103–4107.

[9] Bin Wang, Douglas Brown, Yongsheng Gao, and John La Salle, "Mobile plant leaf identification using smart-phones," in *IEEE International Conference on Image Processing (ICIP)*, 2013, pp. 4417–4421.

[10] Navneet Dalal and Bill Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, 2005, pp. 886–893 vol. 1.

[11] Lim Wei Howe, Farrah Wong, and Ali Chekima, "Comparison of hand segmentation methodologies for hand gesture recognition," in *Information Technology, 2008. ITSim 2008. International Symposium on*, 2008, vol. 2, pp. 1–7.

[12] Xiaojin Zhu, Lei Yang, and Alex Waibel, "Segmenting hands of arbitrary color," in *Automatic Face and Gesture Recognition, 2000. Proceedings. Fourth IEEE International Conference on*, 2000, pp. 446–453.

[13] Jose M. Saavedra, Benjamin Bustos, and Violeta Chang, "An accurate hand segmentation approach using a structure based shape localization technique," in *VISAPP (1)*, 2013, pp. 321–326.

[14] Richard Szeliski, "Image processing," in *Computer Vision: Algorithms and Applications*, chapter 3, pp. 112–113. Springer-Verlag New York, Inc., 1st edition, 2010.

[15] Martin de La Gorce, David J. Fleet, and Nikos Paragios, "Model-based 3d hand pose estimation from monocular video," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 33, no. 9, pp. 1793–1805, Sept 2011.

[16] Constantina Raluca Mihalache and Bogdan Apostol, "Hand pose estimation using hog features from rgb-d data," in *System Theory, Control and Computing (ICSTCC), 2013 17th International Conference*, 2013, pp. 356–361.

[17] Marin Šarić, "Libhand: A library for hand articulation," 2011, Version 0.9.

[18] Jintae Lee and Tosiyasu L. Kunii, "Model-based analysis of hand posture," *Computer Graphics and Applications, IEEE*, vol. 15, no. 5, pp. 77–86, 1995.