

# Hand Gesture Augmented Camera on Google Glass

Toan Tran, Jim Wang, Bell Wang, Mario Malave

**Abstract.** This paper describes the design and implementation of two use cases where the users of Google Glass use their hand gestures to augment camera media and functionality. The first use case is to point the index finger to an object in the scene to highlight the object while taking picture or video. The second use case is to put two hands in a rectangular frame to crop the picture while taking it. We have implemented two hand gesture detectors and one segmentation-based object selection system. The first detector is a Bag-of-Word based SVM classifier using SIFT features. The second detector is based on sliding-window method using SVM classifier with HOG features. The segmentation-based object selection is based on edge detection. Our systems achieve 90% detection rate on hand gestures and 72% recall for object selection. Up to our knowledge, this is the first research report that augments Google Glass camera media and functionality using hand gestures.

## 1. Introduction

The emerging of wearable devices like Google Glass is shifting the paradigm of how we interact with multimedia. With Google Glass, one can take a picture or record a video by voice commands, leaving his/her hands free. The user, hence, can use their hands as an additional interface to the camera.

We want to research and build a computer vision based technology that allows Google Glass users to use hand gestures to enrich the camera media and functionality. For the scope of this class, we research and implement two use cases.

- Using hand pointing gesture to select and highlight objects.
- Using hand-as-aperture to take pictures.

### Select and Highlight Objects using Hand Pointing

When taking video or pictures, the users can point their index finger to an object in the scene to get that object highlighted, e.g. drawing a box around the selected object (Figure 1). For the scope of this project, we limit to highlighting the object after the hand has been withdrawn (i.e. hand is not in the image).



**Figure 1.** Assume the person (left) points at the TV (right), the TV is selected and highlighted.

### Hand-as-Aperture Photography

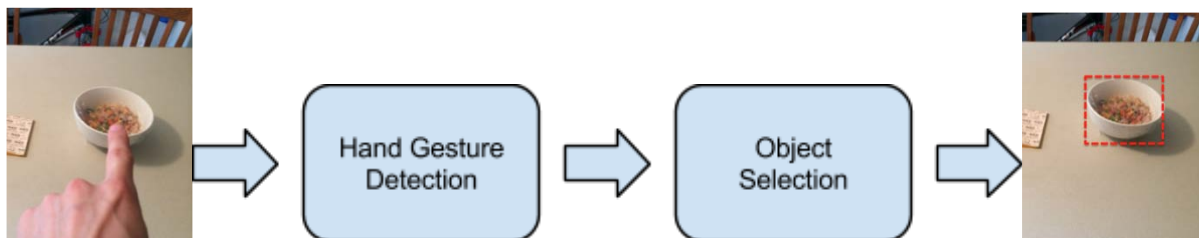
While taking pictures, the users create a hand shape and put it in front of the device camera as an aperture to customize the picture frame. The part of the image enclosed by the hand shape is cropped out as the final image (Figure 2). This project scope is limited to the rectangular hand gesture as Figure 2.



**Figure 2.** Rectangular hand shape is used to select part of the scene to be cropped out.

This project addresses two main problems: hand gesture detection and segmentation-based object selection (Figure 3). One of the main challenges is that general object selection is extremely hard. There are so much variability in real world data, and oftentimes semantic interpretation is required to correctly identify something as an object. State-of-the-art object selection algorithms either relax the problem or use a lot of training data. Other challenges are the requirement for a highly accurate detector of gestures on clutter background and the lack of hand gesture dataset. Our contributions in this project consists of:

- A personalize skin color and texture calibration method based on machine learning
- Two hand gesture detectors based on SIFT and HOG features with detection rate 90%
- A database of 300 hand gestures images
- Object Selection with Edge Detector (OSED) with 72% recall



**Figure 3.** Two main components: gesture detection and object selection

The rest of this report is organized as follows. Section 2 reviews related work. Section 3 describes our solutions to hand gesture detection. Section 4 presents our solution to segmentation-based object selection. Section 5 analyzes the results. The conclusion with future plans are summarized in Section 6.

## 2. Related Work

Hand gesture recognition has been extensively studied for human-machine interaction (HCI) applications. A deep review can be found in [Sarkar 2013]. Many methods in HCI assume that the background is either a simple wall or stable and thus background subtraction can be used to segment out the hands [Chen 2007, Chung 2009, Ren 2010]. HOG, Haar-like, and SIFT features are widely used as feature vectors. HOG yields high detection rate but is not rotation invariant [Freeman 1995, Misra 2011]. Haar-like feature also gives high detection rate but rotation is limited to  $\pm 30$  degrees [Chung 2009, Yun 2009, Chen 2007]. SIFT is one of the most-used feature in case rotation variation are severe [Dardas 2001, Gurjar 2012, Wang 2007]. The work that is most similar to our work is [Dardas 2001] that employed an SVM classifier based on BoW technique and SIFT features. Most of state-of-the-art works just solve the recognition task while our project focuses on detection - classification and localization of the thumb-index corner and fingertip. Fingertip detection was perform using edge detection with thresholding, and local maxima similar to [Hagara 2013, Raheja 2011]. Once the edges are detected, the rows of the image are then analyzed similarly to from top to bottom to identify the local maxima of the fingers.

Skin color and texture detection have also been extensively studied. Different color spaces have been

tested and the most widely-used are HSV, YCrCb, and log-chromaticity for their robust against rotations, scaling, and lighting conditions [Dardas 2001, Khanal 2011, Vezhnevets 2003, Phung 2005]. Gabor filter has been demonstrated a robust one for extracting skin textures [Cula 2003, Jiang 2007, Jiao 2001, Han 2012]. All of these research works learned single thresholds for all people that classify skin pixels and non-skin pixels. Even though skin tone occupies a small localized region on color and texture spaces, a single threshold for all people and illumination seems not an optimal solution. We, therefore, implement a personalized calibration for skin color and texture.

In our initial findings for object selection, literatures discussing methods for image segmentation were studied. The segmentation methods we referred to are *K-means Clustering*, *Mean-shift Clustering*, and *Watershed Segmentation*. For *K-means Clustering* [Mehta 2008], the distance metric in *K-means Clustering* in image segmentation could be the RGB color features and texture features. The main difficulty for K-Means Clustering is to know  $K$  in advance. For *Mean-shift Clustering* [Comaniciu 2002], Comaniciu and Meer describe an algorithm to segment images that are independent of specific models. One good advantage of *Mean-shift Clustering* is that it does not assume data cluster shapes and only requires the window size parameter. However, the window size selection is non-trivial, especially for high dimension feature space. For *Watershed Segmentation* [Meyer 1994], the watershed transform is applied to the modified gradient image where its regional minima occur at foreground and background marker pixels. However, if an image has noise, this will influence the segmentation. Hence, as watershed is created at each minima, *Watershed Segmentation* is highly sensitive to local minima. One of the state-of-the-art object selection and recognition work is that of Arbelaz and Malik [Arbelaz 2011]. Here, the authors present a method to detect contours based on spectral clustering and reduced image segmentation to contour detection in a hierarchical tree. However, their methodology required complexity as well as lots of training images beyond the scope of this class.

### 3. Hand Gesture Detection

We have implemented two different hand gesture detectors for educational and comparative purposes. The first detector is an Bag-of-Word SVM classifier that uses SIFT features. The second one is a sliding-window detector based on an SVM classifier using HOG features. The former is the main detector that is used throughout this project. The latter will be described in Section 3.4. The workflow of the BoW SVM gesture detector consisting of 3 processing components is depicted in Figure 4.

- **Skin Color and Texture Segmentation:** This component classifies each pixel of the input image into either skin (1) or non-skin (0). The color and texture features are described in detail in Section 3.1. Then, connected components of skin pixels are extracted. Components that have size less than 5% of the image size are discarded because they are too small to be a meaningful hand gesture. The output of this component are windows each of which contains a connected component of skin pixels. These windows are called candidate hand windows because they may contain non-hand stuffs that have color and texture similar to those of skin.
- **BoW SVM (SIFT):** This is the classifier that determines if a candidate hand window contains a true hand gesture or not, and what the gesture is. This component is described in detail in Section 3.2. The output of this SVM classifier are windows that are believed to contain true hand and the type of the gesture - either pointing gesture or cropping gesture.
- **Fingertip/CropCorner Localization:** This component receives true hand windows from the SVM classifier and localizes the fingertip or the cropping corner - the corner made by the thumb and the index. The fingertip is detected by finding the top-most hand pixels because we limit our hand pointing gestures to point upward. The cropping corner is described in detail in Section 3.3. The output of this component is the location of the fingertip or cropping corner.

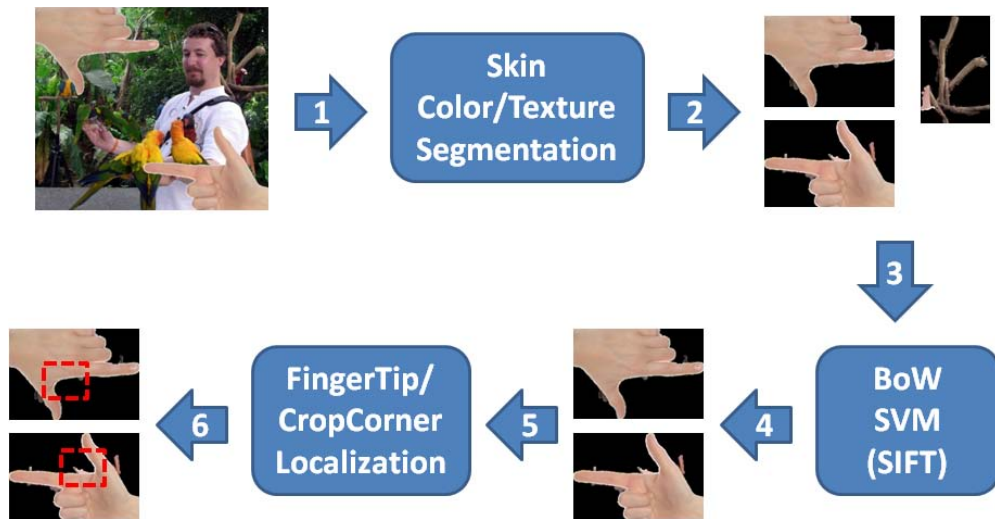


Figure 4. BoW SVM hand gesture detector workflow

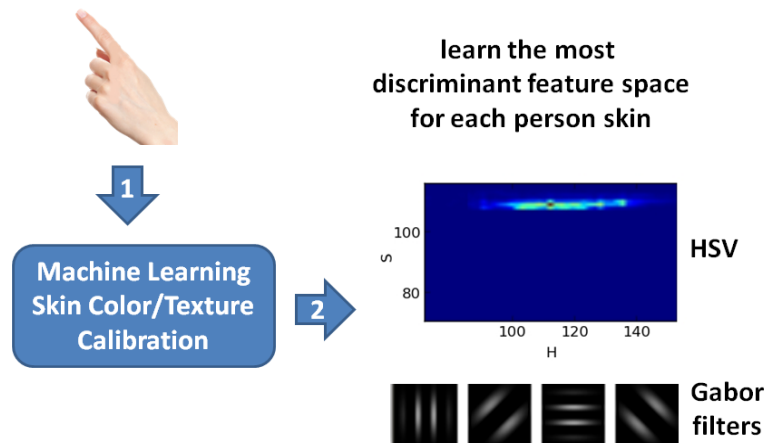
### 3.1. Personalized Skin Color and Texture Calibration

Skin detectors in the literature employ single thresholds for all people. HSV-based skin detectors classify a pixel skin if its  $H$  value is within  $[0, 50]$  and its  $S$  value is within  $[0.23, 0.68]$  [Phung 2005]. Log-chromaticity based skin detectors classify a pixel skin if  $\log(R/G)$  is within  $[0.15, 1.1]$  and  $\log(B/G)$  is within  $[-4, 0.3]$ , where  $R, G, B$  denote red, green, blue values of the pixel [Khanal 2011]. The use of single thresholds incurs the following separation trade-off. If the thresholding range is too small, it cannot cover all skin types. On the other hand, if the range is too big to cover all skin types, it sacrifices the discrimination capability of separating skin and non-skin stuffs. Moreover, it is difficult to adapt to skin change under different illumination conditions. From our experiments, these thresholds yield bad results on our hand collections.

This project also chooses an explicit-thresholding classifier to detect skin pixels for its fast performance, but learns different skin thresholds for different person. When a user runs the app for the first time, his/her hand on a white background is taken. This calibrated hand is used to learn the color and texture features of the user's hand skin. Our machine learning based calibrator learns the most discriminant feature spaces for the user's skin (Figure 5). The input to the calibrator are the user's hand and 100 background images that do not contain skin.

- Skin color calibration in HSV color space: The histogram distribution of  $H$  and  $S$  values of skin pixels is first computed. Then, the calibrator searches for two optimal ranges  $[minH, maxH]$  and  $[minS, maxS]$  that best separate skin pixels from non-skin pixels.
- Skin texture calibration: Texture is computed by 2D Gabor filters. A 2D Gabor filter is a 2D Gaussian function modulated by oriented complex sinusoids. The calibrator searches for an optimal set of Gabor filter parameter, including the scale (Gaussian standard deviation) and the sinusoidal frequency, that best separate skin and non-skin pixels. In our experiment, we use four orientations for Gabor filter, including 0, 45, 90, and 135 degree.

This personalized calibration process is just the first step in our bigger vision: personalized computer vision. Wearable devices such as Google Glass is designed for personal use. We can tailor computer vision apps to each individual user based on the situation and usage pattern. As our next plans, we want to machine-learning-based adjust the calibrated parameters to the illumination change, for example brightness and shading, detected by the light sensor on Google Glass.



**Figure 5.** Personalized skin color and texture calibration

### 3.2. Bag-of-Word SVM Classifier using SIFT Features

Our SVM classifier is based on the standard BoW implementation.

- Training: The first step is extracting the keypoints and SIFT features of hand gesture training images. Then, a k-means clustering algorithm is used to cluster the keypoint descriptors into K clusters and encodes each keypoint by the index of the cluster to which it belongs. Each cluster is considered as a visual word that stands for a particular local pattern shared by the keypoints in that cluster.
- Testing: First, the SIFT keypoints and descriptors of the testing image are extracted. Then, the feature descriptors are matched with the visual words created in the training step. Finally, the generated visual-word vector is fed into the SVM classifier.

### 3.3. Template-based Cropping Corner Localization

Given a hand window, we want to determine the thumb-index corner and two points on the thumb and the index (red points in Figure 6). We observe that the thumb-corner template is quite stable across people and the scale does not vary much because the distance from the user hand to the google glass camera varies very little. So, we opt to use a template-matching technique. Oriented pre-defined thumb-index template are matched across the hand window. We employ pyramid technique to reduce computational time. The hand window is scaled down to 2 pyramid layers. In the second layers, we apply the template matching for every 20 degrees, and limit the range of angle that will be fine-searched in the first layer. In the first layer, we apply template matching for every 5 degrees. So, our matching accuracy is  $\pm 5$  degree.



**Figure 6.** The second pyramid layer template is 4 times smaller than the first layer and thus we can find the match for 18 orientations ( $360/20 = 18$ ). After the second layer, the range of possible angle is shrunk to 20 degrees, so we can apply 4 orientations matching on the first layer ( $20/5 = 4$ ).

### 3.4. Sliding-Window Gesture Detector using HOG Features

For hand cropping gestures, we observe that the thumb-index corner exhibits two following features. First, they are quite stable across people. And second, they are quite a distinct pattern that the background doesn't have. So we decide to train a linear SVM to detect this thumb-index window using sliding-window technique. HOG is used for feature vector. We trained the SVM on 100 thumb-index corner windows on clutter background and 1000 background windows. Each window is 64x64 pixels (Figure 7).



**Figure 7.** The first 3 windows are thumb-index corner on clutter backgrounds. The last 3 windows are background windows.

## 4. Segmentation-based Object Detection

The goal of this section is to select the object that the finger points to. In other words, we wish the draw a bounding box around the object. Here we present the various methods that we tried in sequence.

### 4.1. Initial Findings

**Segmentation.** We first try existing segmentation methods and check if the segmented region can identify correct object. We identified five segmentation techniques to try: K-means, mean-shift, watershed, normalized graph cut, and interactive image segmentation. For K-means, we have tested three cases for  $K$  (e.g. 2,4,6). The algorithms that worked better are k-means, mean-shift, and normalized graph cuts (Appendix Table A.1). These methods could use both color and intensity gradients for determining segmentation. Watershed performed poorly because it works on gray-scale image, but our image in gray-scale is fairly homogeneous. We also tried Grub-Cut, which is an Interactive Image Segmentation method, but we deemed it not satisfactory since it requires the user’s input on both foreground and background (we only have one foreground point, from one finger point). We also experimented on how the number of clusters affect the image segmentation result in the k-means method, and found that the value of  $K$  will be dependent on the heterogeneity of the colors and gradients in the image (see Appendix Table A.1). In general, it is better to over-segment because we can post-process to regroup segments.

**Heuristic - similarity by texture features.** We also tried a completely different approach: start from the known fingertip position and gradually expand outward to mark regions as foreground. Our heuristic is to group similar patches around the fingertip. We built a feature vector based on the texture feature through Gabor filter, then extend the patches around the fingertip until reaching a threshold. However, this method does not work well because it is hard to define an appropriate threshold for different objects. This is because different objects have different threshold among the texture feature similarity and it is impossible to know what the object is in advance and set a proper threshold without doing training.


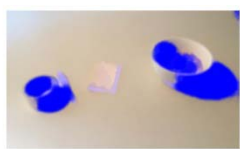


**Difficulties thus far.** One difficulty is to solve the over-segmentation problem once the image is segmented and target object contains several segmented parts. An important challenge is finding the best way to regroup segments together to locate the objects. Another difficulty is to capture the semantic meaning of objects. In reality, it is hard to define what the object is and which object is semantically selected by the user. The accuracy to select an object depends on the definition of an object and the user’s specification. A typical example is that some smaller objects may be contained in some objects with large area, and it is hard to choose which objects is actually selected by the user. The third difficulty is to collect enough data and marking the ground truth for performance evaluation, which were done by human experts in this project.

### 4.2. Relaxed Problem with Assumption

Due to the difficulty of the problem, we decide to introduce an assumption and find an appropriate solution. Our assumption is that user-selected objects are within uniform background and there is no overlapping object. Hence, the relaxed problem is that we want to select an object from a uniform background where no objects are overlapped with each other.

**Baseline - Filter by Background Colors.** In the relaxed problem, our baseline solution is to filter out the

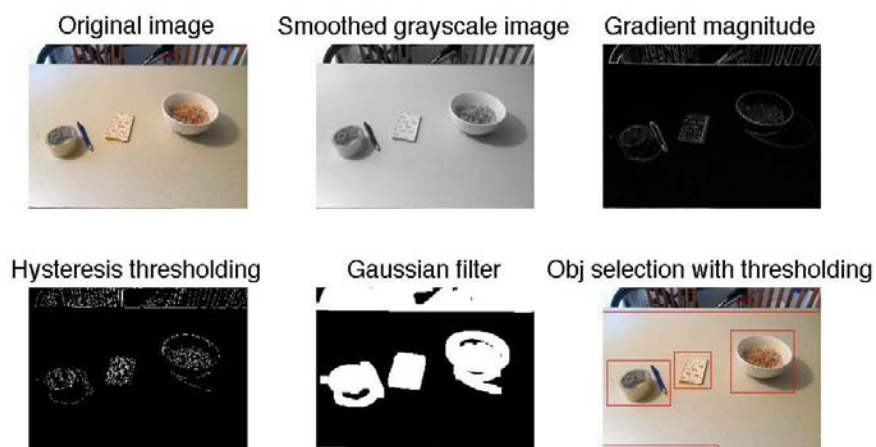
objects from the uniform background color. First, we run 2-means clustering on the whole image based on the color RGB feature and we get two centroids C1 and C2. According to our assumption, the background color should dominate in the whole image, and one of the two centroids should have much more pixels closer to it (C1) than to the other (C2). Then, we choose C1 as an approximate background color and filter out all pixels as part of an object if its Euclidean distance to C1 is less than a threshold. Through tuning, we find that a distance threshold around 100 will provide a decent filtering and all objects can be filtered out. Once we get the filtered pixels, we run connected component algorithms on them to find all connect components (CC). Each CC can be treated as a selected object. The procedure of the baseline can be illustrated in Figure 8.

Original Image	Pixels filtered out	Concatenate Pixels into CCs	Select objects in CCs
			

**Figure 8.** Demonstration of steps in the Naive baseline implementation.

### 4.3. Object Selection with Edge Detection (OSED)

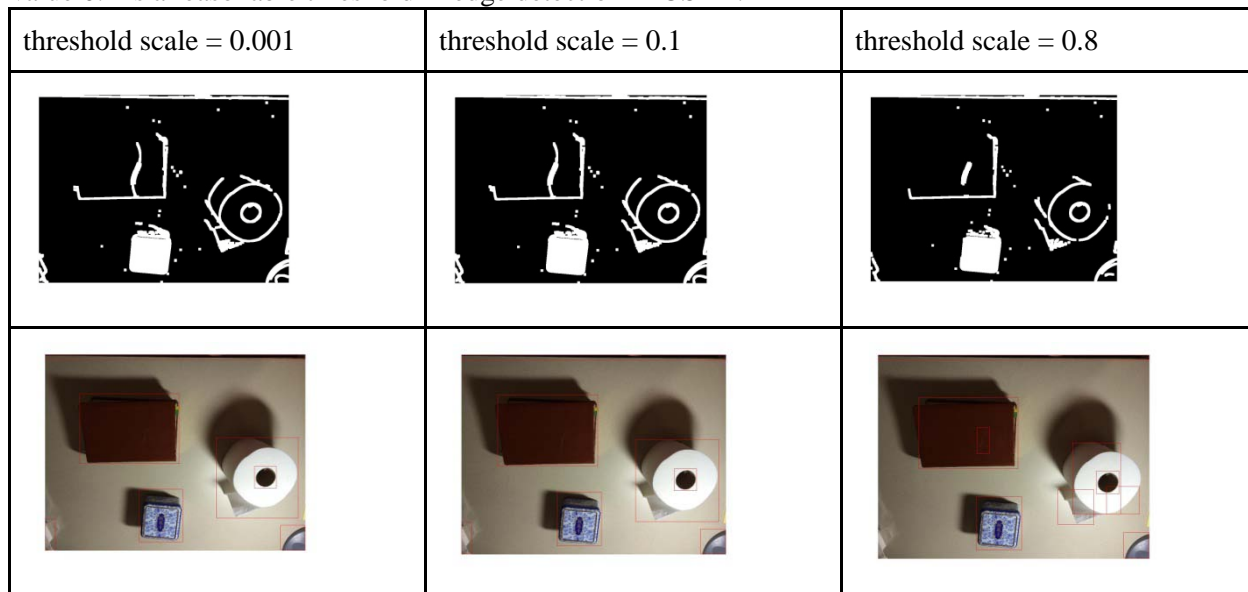
In the relaxed problem, OSED first detects the edges by gradient magnitude and Hysteresis threshold. Then, detected edges are connected via Gaussian filter. The main tuning parameter is the hysteresis threshold. In particular, we use Otsu’s method to define the upper threshold, and set the lower threshold to be a scale factor of the upper. Lastly, we will run connected component algorithms on the connected edges and find connected components (CC) in the image. Each CC can be treated as a selected object. Some CC are small, and we set a patch size threshold to kick out CC that are too small to be deemed an object (the threshold is set such that an image is divided into 500 equal sizes. This is a conservative estimate as the resulting patch is only a dot on the image, and all true objects are larger than this dot. Thus, no parameter tuning was needed here.). The procedure of OSED can be illustrated in the Figure 9.



**Figure 9.** Sample object detection by OSED. Lower threshold is 10% of upper threshold.

In order to analyze Hysteresis threshold on the performance for OSED, we perform evaluations on different Hysteresis lower threshold values and compare the results. We have tried Hysteresis threshold scale factors of 0.001, 0.01, 0.1, and 0.8. We find that there is no great difference between the performance with 0.001, 0.01 and 0.1 scales. Compared with 0.8, we find that with a larger lower threshold value, more objects will be selected in the image. This is because large lower threshold results

in fewer edges being classified. As a result, the edges are more disconnected, so more connected components. As shown in Figure 10, the *stock form* on the right side of the image will have more edges detected with threshold factors 0.001 and 0.1 than those detected with threshold 0.8. For example, the edge detection with threshold scale 0.8 (on the right side) has fewer edges detected which results in more CCs in the complete *stock form* object. Compared with large threshold values, we find that a threshold value 0.1 is a reasonable threshold in edge detection in OSED.



**Figure 10.** Comparison of different threshold scales. Threshold scale is ratio of lower threshold to upper threshold.

## 5. Experiment and Result

### 5.1. Hand Gesture Detection Evaluation and Analysis

We trained the first gesture detector, BoW SIFT SVM on 100 hand cropping images, 100 hand pointing images, and 500 non-hand background images. Our testing data set consists of 50 hand cropping images, 50 hand point images on clutter background, and 100 non-hand background images. First, candidate hand windows are segmented out as discussed in Section 3. These candidate hand windows and 100 testing non-hand background images are fed to the SVM classifier. We get 90% detection rate for hand cropping, 87% detection rate for hand pointing, and 97% for non-hand images. The gesture detection rate is lower than non-hand detection rate because some hand pointing images look like hand cropping and vice versa.

We also trained the second gesture detector, HOG SVM on 100 thumb-index corner windows and 1000 background windows. The detector is then tested on 50 images consisting of the whole hand cropping gestures and 100 background images. We got a detection rate 82% which is quite low because sliding-window technique requires a large training set and is sensitive to rotation.

### 5.2. Segmentation-based Object Selection Evaluation and Analysis

We collect 29 images which contains 106 objects in total. For each image, we manually select all possible objects and treat them as ground truth. Then, we evaluate both the baseline and OSED method (both threshold can filter out patches with size less than 1/500 of the image) in terms of the recall and precision rate (Table 1) based on the ground truth.

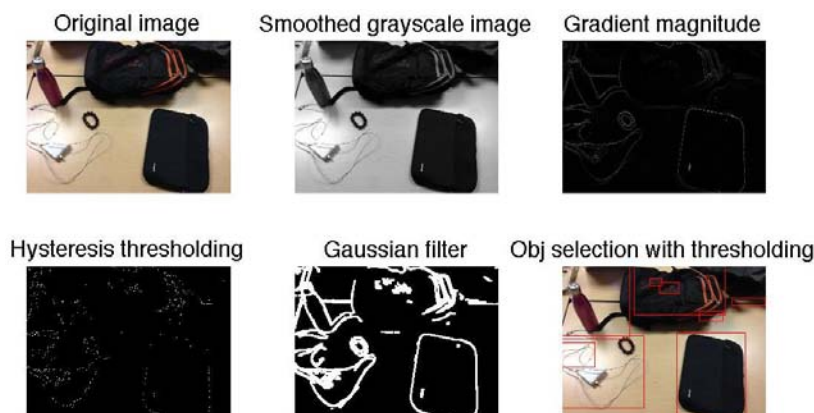
The OSED method achieved a recall of 72% and a precision of 51%, both better than the naive implementation. The relatively high recall suggest that the algorithm does a good job finding the true



positive images. This algorithm is fairly robust in easy scenes and runs efficiently (less than 1 second per image). However, there are also major limitations to OSED. First, shadows can be classified as part of the object; second, edges are not always associated with objects, as can be seen by a lot of stray edges in the Gaussian filter images. Lastly, as demonstrated below, the method is non-robust for complicated scenes.

	Naive	OSED
Precision	37%	51%
Recall	42%	72%

**Table 1.** Result comparing naive baseline with OSED. Here, Naive threshold is set to 100, and the OSED lower:upper threshold ratio is set to 0.1



**Figure 11.** Demonstration of OSED for a complicated scene. We can see that it finds more clusters than necessary and fails to find certain objects (e.g. water bottle).

## 6. Conclusion and Future Work

Our gesture detection methods have demonstrated promising results. We will continue improve the BoW SIFT SVM detector while the HOG SVM is also the best choice for small-rotation single-gesture application. We are considering PCA-SIFT to improve the performance of the detector. We also need to improve our hand segmentation and localization. Currently, we have a database of 300 hand gestures images and plan to collect up to 1000 gestures images using Amazon Mechanical Turk in the next quarter.

For object Selection, OSED have achieved a recall rate over 70% and a precision rate over 50% in the relaxed object selection problem with uniform background assumption. In the future work, we can improve OSED method through tuning the Hysteresis threshold with more data and integrate segmentation methods with OSED method.

Our theoretical design and implementation are ready to be deployed to google glass. We will carry out this project to CS231M - Mobile Computer Vision, in the coming Spring quarter. There we will integrate the gesture detection and object selection into a complete application using Google Glass SDK, and optimize for real-time performance.

## Acknowledgement

We would like to thank Professor Silvio Savarese for discussion about initial project direction. We are also indebted to David Held for insightful suggestions throughout course project. Lastly, we thank Artem Vasilyev for discussions.

## Reference

- [Dardas 2011] N.H. Dardas and N.D. Georganas. Real-time Hand Gesture Detection and Recognition using Bag-of- Features and Support Vector Machine Techniques. IEEE Transactions On Instrumentation And Measurement, 2011.
- [Freeman 1995] W.T. Freeman and M. Roth. Orientation Histograms for Hand Gesture Recognition. IEEE International Workshop on Automatic Face and Gesture Recognition, Zurich, June, 1995.
- [Gurjal 2012] P. Gurjal and K.Kunnur. Real-Time Hand Gesture Recognition Using SIFT. International Journal of Electronics and Electrical Engineering, Volume 2 Issue 3 (March 2012)
- [Misra 2011] A. Misra, A. Takashi, T. Okatani, and K. Deguchi. Hand Gesture Recognition using Histogram of Oriented Gradients and Partial Least Squares Regression. MVA2011 IAPR Conference on Machine Vision Applications, 2011
- [Cula 2003] O.G. Cula, K.J. Dana, F.P. Murphy, and B.K. Rao. Skin Texture Modeling. International Journal of Computer Vision (IJCV) 2003.
- [Jiang 2007] Z. Jiang, M. Yao, and W. Jiang. Skin Detection Using Color, Texture and Space Information. International Conference on Fuzzy Systems and Knowledge Discovery (FSKD 2007)
- [Jiao 2001] F. Jiao, W. Gao, L. Duan, and G. Cui. Detecting Adult Image Using Multiple Features. IEEE 2001.
- [Han 2012] W.Y. Han and J.C Lee. Palm Vein Recognition Using Adaptive Gabor Filter. Expert Systems with Applications 39 (2012) 13225–13234
- [Khanal 2011] B. Khanal and D. Sidibé. Efficient Skin Detection under Severe Illumination Changes and Shadows, In proceeding of: Intelligent Robotics and Applications - 4th International Conference, ICIRA 2011.
- [Sarkar 2013] A.R. Sarkar, G. Sanyal, and S. Majumder. Hand Gesture Recognition Systems: A Survey. International Journal of Computer Applications (0975 – 8887), Volume 71– No.15, May 2013
- [Chen 2007] Q. Chen, N. Georganas, and E. Petriu, “Real-time vision-based hand gesture recognition using Haar-like features,” in Proc. IEEE IMTC, 2007, pp. 1–6.
- [Ren 2010] Y. Ren and C. Gu, “Real-time hand gesture recognition based on vision,” in Proc. Edutainment, 2010, pp. 468–475.
- [Chung 2009] W. Chung, X. Wu, and Y. Xu, “A real time hand gesture recognition based on Haar wavelet representation,” in Proc. IEEE Int. Conf. Robot. Biomimetics, 2009, pp. 336–341.
- [Wang 2007] C.C. Wang and K.C. Wang. Hand Posture Recognition Using Adaboost with SIFT for Human Robot Interaction. Proceedings of the International Conference on Advanced Robotics (ICAR'07).
- [Vezhnevets 2003] V. Vezhnevets, V. Sazonov, and A. Andreeva. A Survey on Pixel-Based Skin Color Detection Techniques. GRAPHICON-2003
- [Phung 2005] 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.
- [Hagara 2013] M. Hagara. Fingertip Detection for Virtual Keyboard Based on Camera. 23th Conference Radioelektronika 2013, April 16-17, 356-360.
- [Raheja 2011] J. Raheja. An Efficient Real Time Method of Fingertip Detection. International Conference on “Trends in Industrial Measurements and Automation” TIMA–2011
- [Arbelaz 2011] P. Arbelaz, M. Maire, C Fowlkes, and J. Malik, A Contour Detection and Hierarchical Image Segmentation, IEEE, 2011.
- [Comaniciu 2002] D. Comaniciu and P. Meer, Mean shift: A robust approach toward feature space analysis (2002)
- [Mehta 2008] A. Mehta and S. Tatiraju. Image Segmentation using k-means clustering, EM and Normalized Cuts (2008)
- [Sobottka 1998] K. Sobottka and I. Pitas. A Novel Method for Automatic Face Segmentation, Facial Feature Extraction and Tracking, Signal Processing: Image Communication 12(3), 263-281 (1998).
- [Meyer 1994] F. Meyer. Topographic distance and watershed lines. Signal Processing, 38(1):113 – 125, 1994.