SINGLE-HAND GESTURE RECOGNITION USING IMAGE/VIDEO PROCESSING AND MACHINE LEARNING TECHNIQUES

Christos G. Bampis [1] *and Jinseok Choi* [2]

[1] Department of Electr. and Computer Eng., University of Texas at Austin, Austin, TX 78712-0240, USA.
[2] Department of Electr. and Computer Eng., University of Texas at Austin, Austin, TX 78712-0240, USA.

bampis@utexas.edu, jinseokchoi89@utexas.edu

ABSTRACT

In this work, we introduce a new method for gesture recognition by combining image/video processing and machine learning methods. Inspired by previous work in the heavily studied field of gesture recognition, we focus on robust and efficient methods that achieve single-hand gesture recognition and propose to easily extend this system to more complex actions and backgrounds. The steps of our method can be summarized to Kalman motion detection, unsupervised color clustering, feature extraction and SVM classification. Our contribution is to achieve single-hand gesture recognition which can be applied to any static camera without the need for any special devices such as a Kinect device [1] or motion sensors [2].

Index Terms— single-hand gesture recognition, Kalman filtering, k-means color clustering, SVM classification, HoG features

1. INTRODUCTION

Gesture recognition can be defined as the process of analyzing and understanding meaningful movements of the hands, arms and face of humans. In particular, the field of hand-gesture recognition has emerged as a promising research area along with the development of visual devices, image/video processing techniques and machine learning techniques. Admittedly, there has been a lot of work in the field (see [3] and [4] for some interesting surveys) and more and more sophisticated models have been proposed. The focus of this work is not to do an exhaustive literature survey so we briefly mention some notable work on this field. For example, Stenger et al. [5] used a hierarchical bayesian filter for hand tracking by using a 3D hand model. Hidden Markov models have also been broadly used in similar tasks like american sign language recognition [6]. Alternatively, skin segmentation methods coupled with an active learning scheme were discussed in [7]. Most approaches have to deal with illumination variations, complex backgrounds and occlusions as well as the different hand postures and poses.

In this work we develop and combine new techniques in the field of hand gesture recognition. We are defining single-hand gesture recognition as our field of work which can be used in many human computer interaction (HCI) applications. We exploit both image/video processing and machine learning techniques to develop a unified recognition system. Our fundamental assumption is that the hand motion is the predominant one, thus we perform a motion detection step in order to limit the region around the hand. Instead of applying some simple background subtraction technique [8], we use the Kalman filter [9] to achieve robustness. For example, in [10] a Kalman filter formulation is proposed, providing the robustness for optical user motion tracking. The Kalman filter that we use in our algorithm outputs a bounding box surrounding the hand by tracking the hand motion. Then, we isolate the target (i.e. the hand) by perform a k-means clustering step based on the Lab colorspace. As usual, we perform some hand mask post-correction steps and then extract HoG features. Finally, we train and test a Support Vector Machine (SVM) classifier using the extracted features to perform our gesture recognition step. Section 2 describes the details of the motion prediction and section 3 discusses the gesture extraction step. In Section 4 we talk about our classification step and sections 5 and 6 discuss computational aspects and some future work respectively.

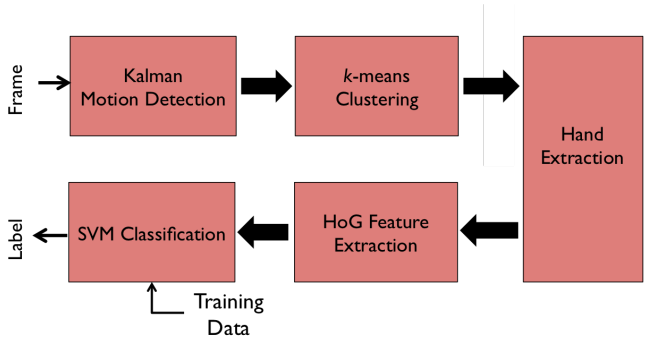

Fig. 1: Overview of the proposed gesture recognition system

2. MOTION DETECTION USING KALMAN FILTER

Our first step in this method, is to apply motion detection by developing a Kalman filter implementation. The Kalman motion detection has the purpose of constraining post processing region to increase the accuracy of the post processes such as k-means clustering, hand extraction and SVM classification by capturing only the largest motion area, which is assumed to be the region of a hand motion, and then the spacial information of this region is sent to the post processes in the form of bounding box. The role of this Kalman filter in our method is very important as it can possibly eliminate false positives in a video frame by limiting the region only around the hand and this increases the robustness of our system as a whole. In addition, this extraction step uses this spacial information from the filter to accurately distinguish the most likely-hand cluster.

The Kalman filter is essentially a set of mathematical equations, which implement a predictor-updater type estimator. The Kalman motion detection algorithm is shown in Fig. 2. This method is often considered as optimal in the sense that it minimizes the estimated error covariance. This filter also coincides well with the implementation of real-time gesture detection as it only needs to keep 1-time previous data. The $N \times 1$ state vector $\mathbf{x}_k$ contains the information of x and y centroid, and width and height of the motion range at time k

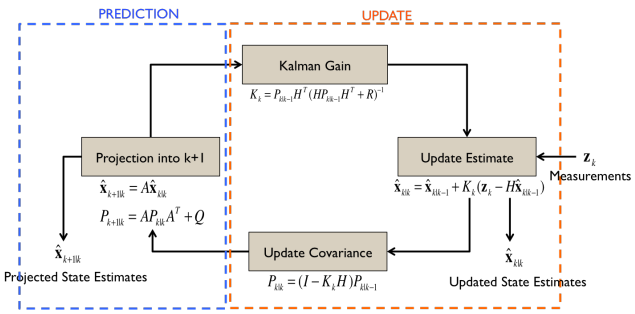

Fig. 2: Motion estimation using Kalman filtering

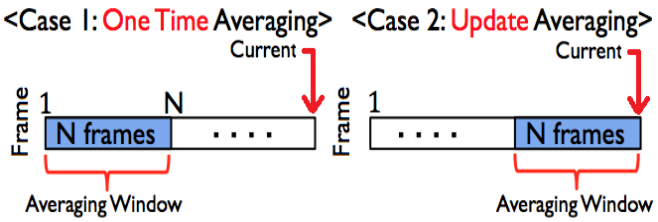

Fig. 3: Two Approaches of Background Calculation

($N = 4$ in our case). This state vector and measurement $\mathbf{z}_k$ can be modeled as:

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + \mathbf{w}_k \tag{1}$$

$$\mathbf{z}_k = H\mathbf{x}_k + \mathbf{v}_k \tag{2}$$

where A is the $N \times N$ state transition matrix of the process from the state at k to the state at $k+1$ and is assumed stationary over time. We set A as an identity matrix in our simulation assuming the little change of the hand position over time. Also, $\mathbf{w}_k$ is the $N \times 1$ vector of the associated white noise process with known covariance, and H is the $N \times N$ transformation matrix that maps the state vector parameters into the measurement domain. We also set H as an identify matrix assuming that the difference between $\mathbf{z}_k$ and $\mathbf{x}_k$ only comes from $\mathbf{v}_k$, and $\mathbf{v}_k$ is the $N \times 1$ vector of the associated measurement error. Based on (1) and (2), we model our motion detection process using the Kalman filter and the prediction part of equations are represented as

$$P_{k|k-1} = AP_{k-1|k-1}A^\top + Q \tag{3}$$

$$\hat{\mathbf{x}}_{k|k-1} = A\hat{\mathbf{x}}_{k-1|k-1} \tag{4}$$

where $\hat{\mathbf{x}}_{k|k}$ means the updated state estimate at time k and $\hat{\mathbf{x}}_{k|k-1}$ is the predicted state estimate at time k with the information of motion centroid and range. Q is the $N \times N$ covariance matrix of $\mathbf{w}_k$. In our simulation, we assume that Q is a time-invariant and diagonal matrix with small values and it shows robustness of the filter performance for the videos where hand-motion dominance assumption is satisfied. Note that $P_{k|k}$ is the mean squared error of $\mathbf{x}_k$ with $\hat{\mathbf{x}}_{k|k}$ and $P_{k|k-1}$ is that of $\mathbf{x}_k$ with $\hat{\mathbf{x}}_{k|k-1}$ i.e.

$$P_{k|k} = E[e_k e_k^\top] = E[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k})^\top] \tag{5}$$

$$P_{k|k-1} = E[e_k e_k^\top] = E[(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})(\mathbf{x}_k - \hat{\mathbf{x}}_{k|k-1})^\top] \tag{6}$$

The results of the prediction process are fed back to the update part of the Kalman motion detection process using the Kalman filter and it is calculated as follows

$$K_k = P_{k|k-1}(HP_{k|k-1}H^\top) + R)^{-1} \tag{7}$$

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + K_k(\mathbf{z}_k - H\hat{\mathbf{x}}_{k|k-1}) \tag{8}$$

$$P_{k|k} = (I - K_k H)P_{k|k-1} \tag{9}$$

where R is the $N \times N$ covariance matrix of $\mathbf{v}_k$ and we assume that it has small valued elements and we set the values based on some empirical results in our simulation. K_k is the Kalman gain, which minimizes $P_{k|k}$. We use the updated state vector $\hat{\mathbf{x}}_{k|k}$ as our detection result forming a bounding box to constrain the processing region for eliminating false positives and increasing the accuracy of the following processes.

In practice, to predict and update the state estimate, we need an accurate measurement $\mathbf{z}_k$, and we propose a simple way of getting the measurement. We capture the difference between a reference frame and a current frame at time k in pixel-wise and we call the reference frame as background, i.e. based on a threshold, we calculate the difference between the background frame and the current frame in each color dimension. If the difference is more than the threshold we set, it will get 1, otherwise 0 and we do entry-wise binary operation (OR) for all color dimension. As a result, all the 1-valued clusters represent motions in a frame, and we pick the largest cluster as the most likely-hand cluster and use it as the measurement $\mathbf{z}_k$. When it comes to colorspace, we use the Lab colorspace instead of RGB colorspace without using L channel since we found the fact that luminance can easily deteriorate the accuracy of this measuring process. As an example, we mention the possible variations in the number of photons captured by the webcam. Thus, we only use a and b channel to compare the background and the current frame with the threshold of 350 in our simulation. This threshold can be adaptively set with respect to the color of the background. In other words, if the background has severe brightness condition or color similarity with a hand, then the threshold needs to be low to capture an accurate motion region and vice versa.

To obtain the background, we use two different approaches and the two approaches are shown in the Fig. 3. In the first case, we compute the background only once by averaging of the first N frames in the video sequence and the background frame is commonly used for the rest of the frames. The immediate advantage of this approach is that it takes small computational resources. However, even a small perturbation of the camera immediately implies that this reference frame is likely to lose its properties (since it no longer corresponds to the background of new frames coming after this movement) so the quality of measurement will get deteriorated. Therefore, we propose a different approach according to which we create the sliding window of length N, i.e. we keep updating the background by averaging the previous N frames from a current frame. As a result, this measuring process can deal with the change of background caused by normal user movement like walking, rotating the camera and zooming in or out. The main drawbacks of this choice are twofold: it takes more computational time as it averages N frames for each newly incoming frame and this could slow down our method, and if a user makes little or no movement over time, the measuring will fail since it will take the hand as the part of background. Given that the solution for this problem is to increase N at the expense of speed once again, we propose a principled choice: for static environments where there is no change of background, we use the first approach and conversely, for the environment were the background possibly changes, we use the second option using a small N (say 20), so that we can accommodate for this dynamic motion environment.

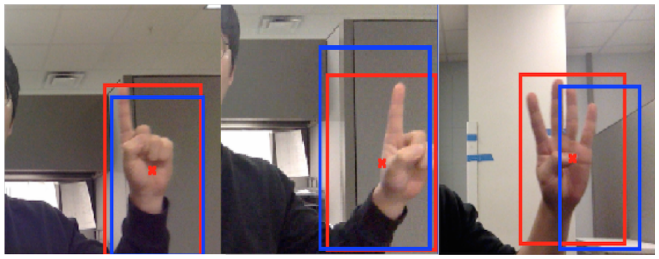

Fig. 4: Kalman motion detection for 3 frames, where the blue bounding box is the prediction and the red one is the actual

3. GESTURE DETECTION AND MASK EXTRACTION

Given the Kalman bounding box region, our next step is to accurately isolate the hand region by exploiting the differences between skin color and its background. To do so, we employ a color clustering on a pixel basis. We know in advance that very simple and fast methods like thresholding and using gray scale or RGB values will typically fail, but at the same time our goal is to employ a real-time technique. In order to balance between these two factors, we employ a k-means clustering on the perceptually uniform Lab colorspace by removing the L component, which represents brightness. As a result, the pixel-wise clustering will be robust to illumination changes and will lay more emphasis on the actual color of objects. For example, Fig. 5 shows an interesting result when one uses only the a and b channels from the Lab colorspace instead of the raw RGB values. Notice the illumination effect in the user's face in the direction of the light when using RGB values. In contrast, using only a and b values succeeds in classifying the human face and hand in the same cluster (gray) by excluding other non-skin objects (black or white).

We now give a very brief description of the k-means algorithm so that the description of our method is self-contained. Given a set of observations $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots \mathbf{x}_n\}, \mathbf{x}_i \in \mathcal{R}^d$, we want to partition them into k $(<= n)$ clusters $S = S_1, S_2, \ldots, S_c$ so as to minimize the within-cluster sum of squares. In other words, the k-means algorithm is expressed as:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{\mathbf{x}_j \in S_i} \|\mathbf{x}_j - \mu_i\|^2 \tag{10}$$

In practice, we alternate between two steps the assignment step and the update step similarly to the EM algorithm. The update step computes the cluster centers given cluster memberships and the assignment step assigns every point to the cluster whose center is closer to in the euclidean sense. Note that we start the whole iterative process by simply randomizing cluster centers and repeating the two steps for 100 iterations.

Choosing the right number of clusters in k-means is application dependent and so we experimented with some small values for $k \in \{2 \ldots 4\}$. We picked $k = 2$, since we could observe better results for this value. Note that that the k-means is an unsupervised method and hence we cannot know which of the resulting two classes is actually the hand region that we wish to extract. We note that color classification is a possible solution as shown in Fig. 6 where we can see that since large a and b values represent skin regions we can classify the red points (skin) vs the blue ones (non-skin). However, we note that this experiment was carried out in a relatively constrained lightning environment and such an observation will not generalize. Instead of performing some color classification using skin cluster centers, we further assume that the hand region is the one that is central to the bounding box we got from the Kalman filtering.This observation is in agreement with our original assumption i.e. motion is mostly due to hand motion and, as a result, the Kalman filter will be giving a bounding box which is likely to be located mostly around the hand region. To do so, we compute the average distance (i.e. divided by each region's size) between every point and the bounding box's center. The compactness of the hand region and the size invariance of our distance metric ensure that the region that has the smallest sum of average distances is the hand one. We also note that we originally used the distances between every point and its respective region center as a decision criteria which proved to be inaccurate for compact backgrounds.

Fig. 5: k-means for RGB vs. Lab values after removing L channel, $k = 3$

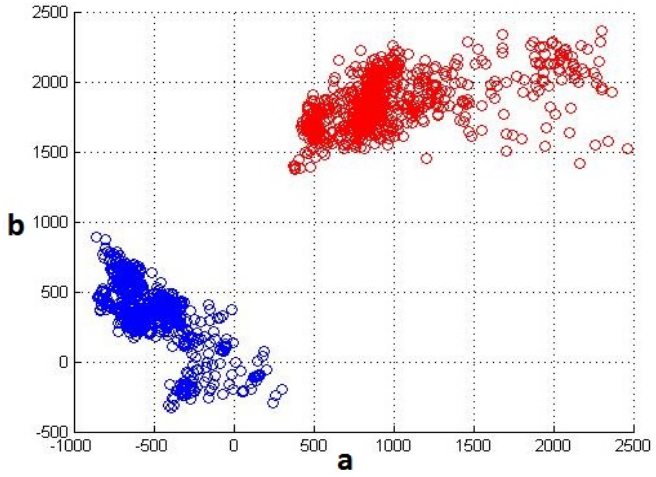

Fig. 6: Cluster centers using our training image dataset

Furthermore, we propose to address some of the drawbacks of the k-means method (e.g. sensitivity to initialization and number of iterations to converge), by replicating some k-means instances for a small number of iterations and picking the best one, i.e. the one that produces the smaller sum of distances. In practice, this was computationally trivial and relatively unimportant since we did not note a big difference in the extraction results, mainly due to the constrained region that the k-means is applied to. We also need to account for false registrations in the resulting Kalman box, i.e. we should refine the bounding box by keeping only the main connected component in the resulting region by assuming that this compo-

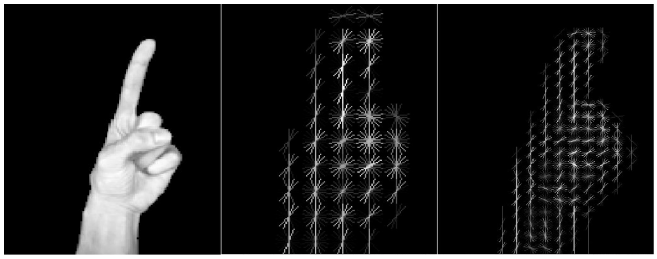

Fig. 7: HoG features for 9 orientations, cell size 8 and 16 respectively, original image resized for display

nent will correspond to the hand region. As a result, we perform a largest connected component analysis and set other components to zero. This proved very important when the user's hand was close to his head or there were some skin colored objects even inside this smaller region, since we could now remove them efficiently. As a further step to our gesture recognition system, we effectively impose a spatial regularity by a morphological filling procedure to remove holes from the hand region. Applying morphological opening and closing using a small structuring element is widely used in many other methods that rely on the extracted mask. In our case, we considered this to be less important since we are relying on the HoG features (see next section).

4. GESTURE CLASSIFICATION

After extracting the hand region we apply a uniform background on every test frame and then perform gesture classification. Support Vector Machines have gained a lot of popularity as supervised learning classifiers. In this work, we used the LibSVM [11] library to train and test our system. Luckily, LibSVM supports multi-class classification using the well-known ones vs. all method. For every image (training or testing) we apply feature extraction on the grayscale versions of every test frame. We used Histogram of oriented Gradients (HoG) features [12] which are broadly used in object detection tasks with a cell size of 16 by 16. This type of features captures gray image gradients in different orientations after performing a histogram binning scheme across small image cells. A key point of the HoG features computation is that gradient strength normalization is performed locally which reduces the effect of illumination and contrast.We then trained our training model offline using a radial basis function (RBF) kernel to perform non-linear classification where the soft margin parameter C and the scaling parameter $\gamma = 4e-04$ were set using a 5-fold cross validation scheme on a relatively large search grid. The training dataset included 747 images captured manually using an Iphone 5S device and resized to match the laptop's webcam resolution that we used.

We used a set of 6 different labels from 0 to 5 denoting different single hand gestures, as shown in Fig. 8. Our dataset includes 5 different subjects performing these gestures under different scaling, orientation and illumination conditions, so that we can more accurately capture these variations in the testing scheme. We ensured that all labels are fairly represented in the training dataset and that different hand sizes are also accommodated. All features were normalized using their Z scores before being fed to the classifier. Our test frames were of varying size (equal to the size of the Kalman bounding box) whereas our training images were 164×123. As a result, we experimented on two different approaches. The first one was to simply resize the test frames to match them with the training ones, so that our features are of constant size. Despite the

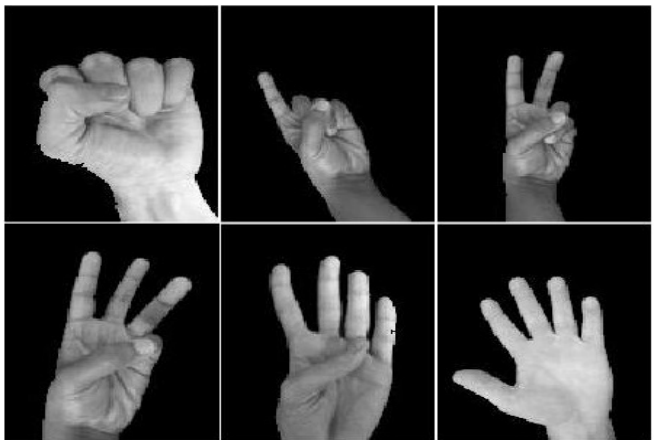

Fig. 8: Representative training images

underlying scaling of our features and the fact that HoG is not scale invariant, we produced good classification results, because of our normalization scheme and since the resizing factor was most of the times not very large. Our second approach was to use SIFT [13] features instead of HoG combined with a Bag of Words (BOW) model and a vector quantization (VQ) step [14]. Briefly speaking, this method aims to keep the strongest features in the data and then build a vocabulary of 500 codewords where every feature is being assigned to. Then, the final feature vector is the histogram of values for this codebook. As usual, we removed outlier images from our training dataset and partitioned it randomly to 30% training and 70% validation to measure the offline accuracy. Unfortunately, this approach worked nicely only in this offline procedure. We believe that this is due to some imperfection in how our VQ actually can capture these variations for the test frames. Due to time constraints, we leave this investigation for future work.

Some experimental results can be seen in Fig. 9. Note that since the Kalman motion detection step reduces the area that we are feeding to the k-means and the SVM classifier, we can still get fine results even for complex backgrounds with multiple skin-colored objects, such as other people's face or hands. Our underlying assumptions are that no extreme lightning or sharp motions take place which can be thought of as a limitation to our work. However, even when these severe conditions are present, our method (despite its simplicity) partially accounts for them. For example, say that some big motion takes place in the background, like a person walking behind the user. Then, the Kalman motion detection will detect this motion and the gesture recognition will fail. However, given that this motion does not persist in the background we only need to wait for the background stack to get rid of this effect and then the user can continue to use the system. In addition, lightning conditions can be addressed effectively by modifying the thresholding parameter as mentioned before.

5. COMPUTATIONAL ASPECTS

Our main goal is to apply this recognition system in real-time. The main bottlenecks of our system are the slow processing power of the Matlab software implementation that we used. Although we have written most of the project in OpenCV using C++, due to time constraints we do not present final results using it. However, our algorithmic steps are easy to implement in all programming environments, like an Android application. We note that there is a time

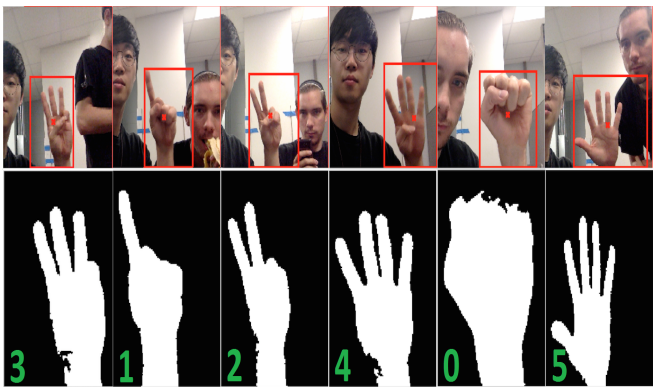

Fig. 9: Test frames and their classification labels

lag between two consecutive frames i.e frame processing lag and Kalman background computation lag. As a result, we do not capture frames at the webcam's maximum frame rate. In real-life applications, we have to take this effect into account since the hand motion is continuous hence a real-time frame processing must be in place. Fortunately, our experiments in OpenCV showed that this effect is dependent on the slow high-level programming nature of Matlab and thus can be addressed effectively.

6. CONCLUSION

In this project we introduced a novel method to detect, extract and classify single hand gestures. We focused on developing new ideas that can effectively address illumination effects, hand shape and size variations. For this reason, we exploited the robustness of the Kalman filtering, the speed and efficiency of the k-means clustering and the effectiveness of HoG features with an SVM classifier. We created a single hand gesture dataset that can capture most variations in real life scenarios, such as rigid transformations and lightning conditions. However, we still need to further improve our system in terms of background complexity and classification accuracy. We propose further investigation on how to improve our features using SIFT combined with the VQ method. In addition, a full OpenCV implementation will further exploit the merits of our work and make an Android application a natural step further. Finally, we intend to include an extended set of single hand gestures which express a wider set of actions.

7. REFERENCES

[1] Z. Ren, J. Yuan, J. Meng, and Z. Zhang, "Robust part-based hand gesture recognition using kinect sensor," *Multimedia, IEEE Transactions on*, vol. 15, no. 5, pp. 1110–1120, 2013.

[2] R. Xu, S. Zhou, and W. J. Li, "Mems accelerometer based nonspecific-user hand gesture recognition," *Sensors Journal, IEEE*, vol. 12, no. 5, pp. 1166–1173, 2012.

[3] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 19, no. 7, pp. 677–695, 1997.

[4] S. Mitra and T. Acharya, "Gesture recognition: A survey," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 37, no. 3, pp. 311–324, 2007.

[5] B. Stenger, A. Thayananthan, P. H. Torr, and R. Cipolla, "Model-based hand tracking using a hierarchical bayesian filter," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 9, pp. 1372–1384, 2006.

[6] T. Starner and A. Pentland, "Real-time american sign language recognition from video using hidden markov models," in *Computer Vision, 1995. Proceedings., International Symposium on*, Nov 1995, pp. 265–270.

[7] H. Francke, J. Ruiz-del Solar, and R. Verschae, "Real-time hand gesture detection and recognition using boosted classifiers and active learning," in *Advances in Image and Video Technology*. Springer, 2007, pp. 533–547.

[8] S. C. Sen-Ching and C. Kamath, "Robust techniques for background subtraction in urban traffic video," in *Electronic Imaging 2004*. International Society for Optics and Photonics, 2004, pp. 881–892.

[9] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Fluids Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[10] K. Dorfmüller-Ulhaas, "Robust optical user motion tracking using a kalman filter," in *10th ACM Symposium on Virtual Reality Software and Technology*. Citeseer, 2003.

[11] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

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

[13] D. Lowe, "Object recognition from local scale-invariant features," in *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, vol. 2, 1999, pp. 1150–1157 vol.2.

[14] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 2, 2006, pp. 2161–2168.