Hand Gesture Recognition with Batch and Reinforcement Learning

Arpit Goyal ARPITG@VT.EDU
Andrew Ciambrone ANDRJC4@VT.EDU
Hossameldin Shahin HSHAHIN@VT.EDU

Abstract

In this paper, we present a system for real-time recognition of user-defined static hand gestures captured via a traditional web camera. We use SURF descriptors to get the bag-of-visual-words features of the user's hand, and use these features to train a multi-class supervised learning model. We choose the best learning model from (SVM, Neural Networks, Decision Trees, and Random Forests) and the best model parameters using hyper-parameter optimization algorithm. During test time, we use these bag-of-visual words features to predict the users hand gestures. The user has the ability to give positive or negative feedback for every prediction to the system, and the system updates itself during test time for better accuracy.

1. Introduction

1.1. Motivation

A Natural user interface, or NUI, is a type of user interface that uses a users natural abilities such as speech or body movements as modes of communication with a computing system (Wigdor & Denis, 2011). It is believed by many that NUI's will revolutionize the way a user interacts with a system because this type of interface is invisible and can be intuitive to the user, as opposed to interfaces that involve contemporary keyboards, mice, touchscreens or joysticks. One common mode of interaction is via hand gestures. By leveraging their hand gestures, users can interact with a system very effectively and intuitively, as it provides a rich mode of communication. Hand gesture recognition is an advancing field that finds its usefulness in robot navigation, HCI, automated homes, virtual games, augmented reality, wearable devices, and various other applications.

Virginia Tech CSx824/ECEx424 final project report.

1.2. Problem

Hand gesture recognition is a problem of classification. A gesture recognition system must be able to read in inputs and predict the correct gesture. Because gesture recognition typically uses a camera to observe the gestures, this problem falls into the fields of computer vision for data extraction and machine learning for classification. Capturing and recognizing the hand gesture is a complex problem that typically requires multiple steps.

1.3. Approach

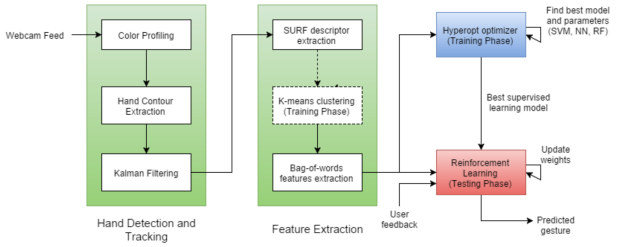

Figure 1. System Overview

Our approach to the problem could be broken into three main steps. The first involves feature extraction. In this step, we do a set of operations on the input webcam feed to detect and track the hand, and finally extract the bag-of-visual-words features from the hand. We use a user-in-the-loop approach to find a robust foreground hand segmentation, as described in section 3.1. Once the blob corresponding to the hand is found, we define a bounding box around it as our region of interest (ROI). We smooth the tracking of the ROI by using a Kalman filtering. This Kalman filter reduces the susceptibility of our system to the noise in foreground segmentation.

Once we have smooth tracking of the ROI, we use SURF descriptors from within this ROI to find 'k' visual words using k-means clustering algorithm. We then quantize the SURF descriptors of each input frame into these 'k' visual words and get the bag-of-words (BOW) features representing the frame.

We train our system using a supervised learning model, like

a multi-class one-vs-rest Support Vector Machine or a Neural Network. We use the BOW features of 100-500 valid frames of each gesture as our training data. We validate the frames by putting thresholds on the minimum number of SURF descriptors in the frame and maximum number of convexity defects in the hand contour. We then optimize and compare the performance of different supervised learning models for our use-case, which we show in section 4.3. The trained supervised learning model gives us an estimate of the initial weight vectors (θ) for the reinforcement learning model.

During test time, we extract the BOW features for each input frame of the webcam feed. We calculate the Q-value Q(s,a) of each gesture prediction (a) for the input (s), using the weight vectors (θ) of the gesture labels. For every BOW feature state (s) the gesture with highest Q-value is predicted. For the system to improve its accuracy during test time, we give the user the ability to give negative/positive feedback for the previous 10 predictions of the system. Based on the user feedback, the system updates the weight vectors (θ) of the gesture labels, using a reinforcement learning model.

2. Related Work

Gesture recognition is not a new idea and a great deal of research has been done in the area. One approach for gesture recognition with traditional webcams is to use cues like the hand contour geometry or the number of convexity defects in the contour to predict the hand gesture. However, this works only for very simple hand gestures. Another approach is via template matching (Khaled & Ali). For this, one needs to save all the training templates for matching during test time, and it clearly falls short for real-time applications.

One popular approach for hand gesture recognition is to use Haar-like features with AdaBoost learning algorithm (Chen & Petriu, 2008). While this approach performs relatively well, it requires a very long training time and a huge data set to train on. Also, it is relatively harder to update the weak classifiers during test time based on user feedback, which is essential to our system. Some approaches do use bag-of-words features to classify a fixed set of gestures using a classifier pre-trained on a huge data set (Dardas & Georganas, 2011). This one-for-all strategy takes away the ability of the user to personalize his gestures. Also, if the classifier predicts wrong gestures, there is no way for the user to give his feedback to the classifier and update the system. This calls for a better strategy to tackle this problem.

Similarly, for hand segmentation and tracking, some approaches use fixed skin color profiling (Dardas & Georganas, 2011) or active contours (Jang & Moon, 2007). However, we found the fixed color segmentation to not perform too well with different users and backgrounds, and active contour method to be very slow for our real-time application.

Because gesture recognition is a classification type problem many types of learning models can be used to solve it. Some of the most common models being used are variations of the SVM like discussed in (Wu & Huang, 1999). However, other learning models can be used to train the system as well.

Researchers from University of Waterloo and MIT (Bergstra & Cox, 2013) have devised an algorithm to optimize and find the best machine learning algorithm for the data being used. Hyperopt provides an library of algorithms and infrastructure for performing hyper-parameters optimization in Python.

3. Technical Contribution

3.1. Feature extraction

We propose a novel approach for feature extraction. For hand segmentation, we initially provide a set of fixed windows to the user on the webcam feed. The user is asked to fit his hand over these windows. On triggering the system, the user's hand color profile is extracted from all these windows. The binary foreground masks extracted from all these windows are added and a median blur is performed along with other morphological operations to get a robust foreground segmentation of the hand even in cluttered background.

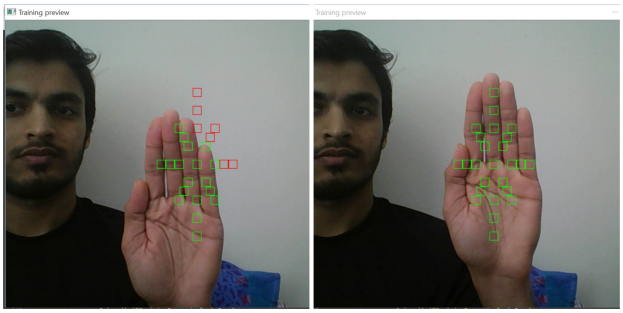

Figure 2. Set of windows presented for hand color profiling

Once we find the blob corresponding to the user's hand, we smoothen the tracking of our region of interest (ROI) using four linear Kalman filters (two for x-and-y coordinates of the top-left corner of the bounding box of the hand, and one each for the height and the width). This reduces the susceptibility of our system to noise in foreground segmentation. We then extract SURF descriptors from the ROI falling

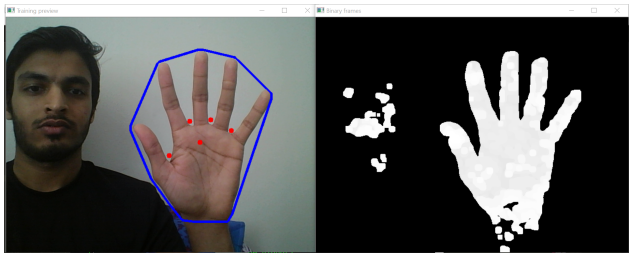

Figure 3. Hand foreground segmentation

within the hand blob being tracked. We experimented with various descriptors like SIFT, SURF, ORB etc. We finally chose SURF descriptors for our system because they provide scale and rotation invariance, and are very efficient to compute (Juan & Gwun, 2009). SURF descriptors are also reasonably resistant to illumination changes, which make them a good choice for our system.

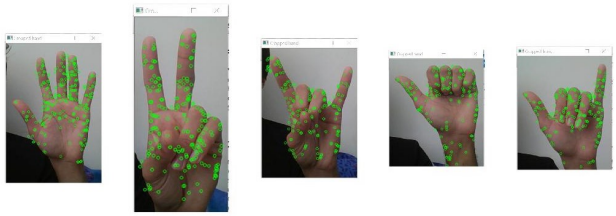

Figure 4. SURF descriptors from different gestures

During training of our system, we build the visual vocabulary by clustering the SURF descriptors from all the training frames into 'k' visual words using k-means clustering algorithm. Once we get our visual vocabulary, we quantize the SURF descriptors for every input frame into these 'k' bins to get the bag-of-words (BOW) features for that frame.

3.2. Supervised Model Training

We use a supervised learning model to train on the BOW features data set from the training frames. (Caruana & Niculescu-Mizil, 2005) provides an empirical comparison of different supervised learning algorithms. For our classification problem, we considered four supervised learning models, namely: support vector machines (SVM), neural networks, decision trees, and random forest. We examined three kernels for the SVM: radial basis function (RBF), linear and polynomial.

Each supervised learning model considered has multiple parameters to optimize which are called hyper-parameters. The SVM has the parameters: regularization constant C, round-off error ϵ, gamma γ, and the degree of the polynomial for polynomial kernels. The decision tree has the parameters: minimal number of data instances at a leaf node, the pruning strategy, and the split strategy (best, random). For the random forests, in addition to decision tree parameters, we have number of decision trees (n estimators). Finally, the neural network has the parameters: number of hidden layers, number of hidden units, and the learning rate α.

3.2.1. Hyper-parameter Optimization

It is not feasible to perform exhaustive search for the best model and the best parameters. Instead we used hyper-parameters optimization method which will search the space of all possible models, model parameters, and any data pre-processing technique to get the best performance (PCA, normalization etc).

One of the effective hyper-parameters optimization algorithms for function minimization is sequential model-based optimization (SMBO), also known as Bayesian optimization. SMBO is ideal for optimizing machine learning supervised models that have many parameters. We used the Hyperopt library (Bergstra & Cox, 2013) which provides algorithms and parallelization infrastructure for performing hyper-parameters optimization.

We implemented an objective function which returns the Hamming loss score of the prediction. Hyperopt minimizes over this objective function to search the optimum model selection, hyper-parameters, and pre-processing of the data.

Since the goal is for each user to train the system for his own selection of hand gestures, the system has to train on different data sets for different users. Because we don't want to assume one configuration to be optimal for all data sets we designed the system to search for best model and best parameters for every training setting. We optimize the hyper-parameters of each of the four learning models and use the best fitting model with optimum parameters to estimate the initial weight vectors (θ) of the reinforcement learning model.

3.3. Reinforcement Learning

Reinforcement learning is the idea that by taking an action a system will receive a level of reinforcement either positive or negative. In our system positive reinforcement comes from the user when the model selects the correct gesture and vice versa. Applying reinforcement learning to a classification problem such as gesture recognition is not intuitive. In the typical sense the reinforcement-learning model contains set of states and a set of actions and a set of rewards (Kaelbing & Moore, 1996). For our system, we use the BOW feature vector of the input frame as state and the system gesture label prediction as the action. Our rewards system is determined by whether or not the model predicted the gesture correctly.

Our initial approach was to try Q-learning algorithm. After much research into the problem we discovered that Q-learning in its pure form may not be suitable for the problem we are trying to solve. Q-learning looks for the action with the most reward. However, it also looks forward one state predicted by the next action to determine the greatest reward at that next state. While this is great in theory it doesn't work for the problem we are attempting to solve since our system is not purely autonomous; i.e. it depends on the input gesture from the user. There are a few publications out there where Q-learning is used for classification such as in (Wiering & Schomaker, 2011) where they treat the pixels as the states and use a binning technique to determine the next action. However implementations like these do not use the trained weight vectors from a supervised learning model like SVM or Neural Network. Similarly, (Lagoudakis & Parr, 2003) uses a variant of Q-learning and approximate policy iteration based on rollouts for classification problems. They use pure supervised learners like SVM in the inner loop of policy iteration algorithm, for autonomous tasks like pendulum balancing and bicycle riding. For our system, we decided to use the following variation of Q-learning algorithm for weight updates and Q-value estimation.

Initially, the weight vectors trained by the supervised learning model are used to get Q-values $Q(s, a)$ of each gesture prediction (a) for every input BOW feature state (s).

$$Q(s, a) := \theta_1(a)x_1(s) + \theta_2(a)x_2(s) + ... + \theta_d(a)x_d(s)$$

In the above equation $\theta(a)$ is the weight vector corresponding to a gesture label 'a', while $x(s)$ is the BOW feature vector for the input state 's'. The Q-values for all the gesture predictions are scaled between 0 and 1 using a squashing function (for instance, logistic function) and the gesture label with the highest Q-value is predicted by the system. We use the difference between the highest Q-value and the second-highest Q-value as a confidence score of the system. If the confidence score is lower than a threshold, then no gesture is predicted by the system.

During test time, the previous 10 predictions are presented to the user on the side of his webcam feed. The user can review and give positive/negative feedback on these side frames using a mouse click (left-click for negative feedback and right-click for positive feedback). The system uses the below equation to update the weights $\theta(a)$ based on the user feedback.

$$\theta_i(a) \leftarrow \theta_i(a) + \alpha(R - Q(s, a))x_i(s)$$

In the above equation, α is the learning rate which ranges between 0 and 1. R is the reward for positive and negative feedback. We use a constant positive value R^+ for positive feedback and a constant negative value R^- for negative feedback. These updated weights are then used to find Q-values for the next incoming frames.

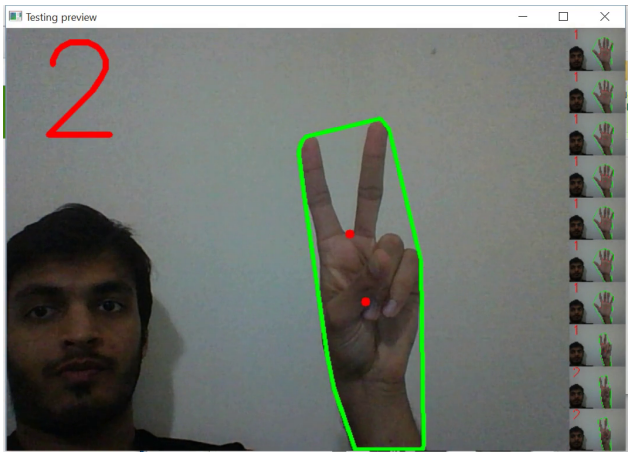

Figure 5. Test Phase. System gesture label prediction is on the top-left corner. The previous 10 predictions are shown as side frames on the right for user feedback.

4. Results and Analysis

In this section, we show the performance of our system with different parameter settings and learning models.

4.1. Data Used

We experimented with various online data sets for our system. Since the goal is for personalization of user gestures, we tried to closely imitate our use-case by gathering data ourselves in real-world settings. This also enabled us to validate the data frames on the number of descriptors present and the number of convexity defects in the hand, to avoid spurious data. We extracted the 300 valid frames for each gesture in 3 different backgrounds and 3 different lighting conditions, with 3 different users. We collected two such data sets, one for 5 gestures and the other for 10 gestures. We divided the collected data set into training, cross-validation and test sets by the ratio of 60:20:20. We use the training set for training the supervised learning models, cross-validation set for hyper-parameters optimization and test set for finally getting a measure of the performance of our system. It should be noted that since our training data sets are relatively small, the results may differ for larger data sets.

4.2. Visual Vocabulary Size

First we explore the performance of our system with change in the number of visual works 'k', without opti-

mizing for the hyper-parameters. We observe that as we increase the vocabulary size, we get better accuracy on the training data. We report the leave-one-out cross validation scores for linear SVM kernel with variation in the number of visual words in Figure 6.

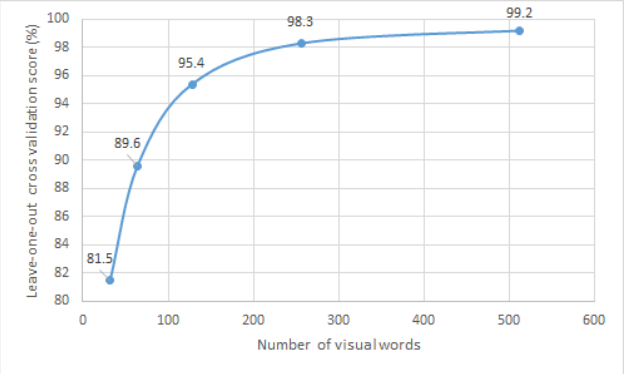

Figure 6. Leave-one-out scores for different vocabulary sizes

While increasing the number of visual words boosts the system training accuracy, it tends to overfit the training data. We found that the test accuracy initially increases with increase in visual vocabulary size, reaches a maximum test score and then starts decreasing. For our application, we found the optimum value of vocabulary size 'k' to lie between 150 and 300.

4.3. Supervised Model Optimization

4.3.1. Hyper-parameters Analysis

Here we analyze the search space for choosing the best parameters using parts of the search space for each model. Noting that since the user will train his own models, it is possible that the best model parameters differ from the presented best model for other training settings.

The search space for hyperopt optimization for SVM with RBF kernel is shown in Figure 7 as an example of the performance of hyperopt. The figure shows the results from 5 gestures data set. Instead of an exhaustive search for the best combination of C and gamma, the algorithm moves in the log space with significantly less number of function evaluations. The algorithm was able to find the area of interest (the blue area with the lowest prediction error) in only 300 function evaluations for C $\in [10^{-5}, 10^{5}]$ and gamma $\in [10^{-10}, 10^{-4}]$.

Figure 8 shows the accuracy of the Decision Tree model for split strategies best and random, using different maximum depth. Result is shown for 5 gestures and 10 gestures data sets. Maximum depth greater than 15 produces comparable prediction error. There is no overall best split strategy for larger maximum depth according to Figure 8.

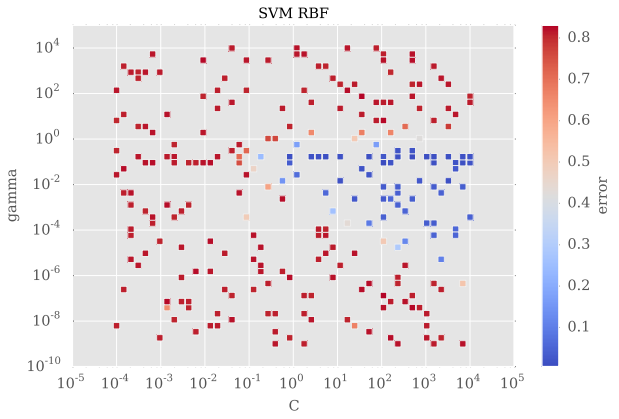

Figure 7. Hyperopt search space for C and gamma parameters of the RBF SVM. The color shows the error.

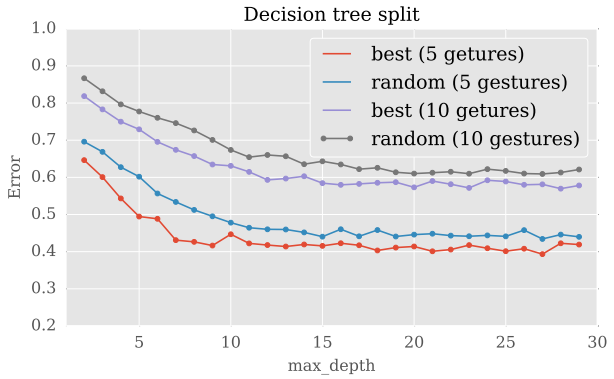

Figure 8. Decision tree performance for split strategies: best and random, with different maximum depth. Results is shown for 5 gestures and 10 gestures data sets.

Random forest accuracy improves by increasing number of estimators to some extent where the improvement is limited compared to the increased running time. Figure 9 shows the prediction error for number of estimators 10, 20, 30 and 40. The effect of maximum depth of the trees within the random forest has a small variable effect after depth of 20. Choosing a general optimal value for the depth is not possible but a suggested range would be from 20 to 40.

4.3.2. Best Models Comparison

We compare the prediction accuracy of the best models chosen by hyperopt optimization for the two data sets: 5 gestures and 10 gestures. Table 1 shows the optimal parameter values achieved by the hyper-parameters optimization algorithm. SVM and Neural networks performed slightly better than decision tree and random forest as can be seen in Figure 10.

Table 1. Best chosen parameters for each model.

Number of Gestures	SVM				Decision Tree		Neural Networks		Random Forest	
	kernel	C	degree	gamma	splitter	max depth	alpha	hidden layers sizes	max depth	n estimators
5	poly	0.72	4	51.80	best	26	0.14	25,27	20	22
10	poly	0.02	4	6.25	best	17	1.67	28,24	13	10

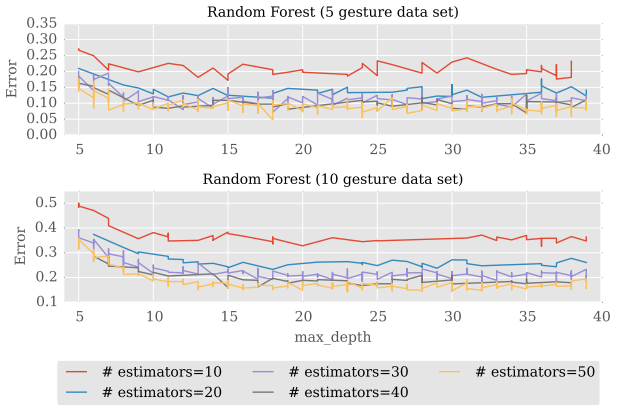

Figure 9. Random forest performance for different number of estimators. split strategies (best and random), and for different maximum depth. Result is shown for 5 gestures and 10 gestures data sets.

Out of all the supervised models considered, the polynomial kernel SVM proved to best fit our data sets (both for 5 gestures as well as 10 gestures). Figure 11 and Figure 12 show the confusion matrices for the two data sets with the polynomial kernel SVM.

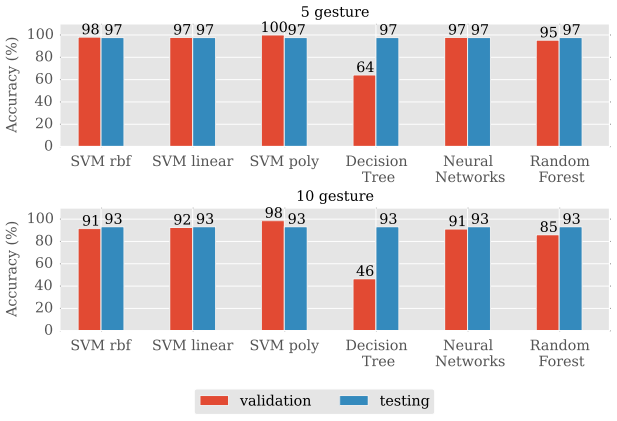

Figure 10. Comparison of the best parameters for each model.

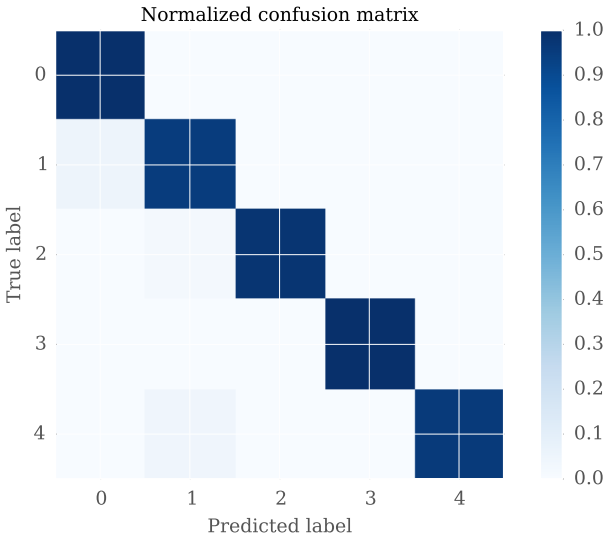

Figure 11. Confusion matrix for 5 gestures data set.

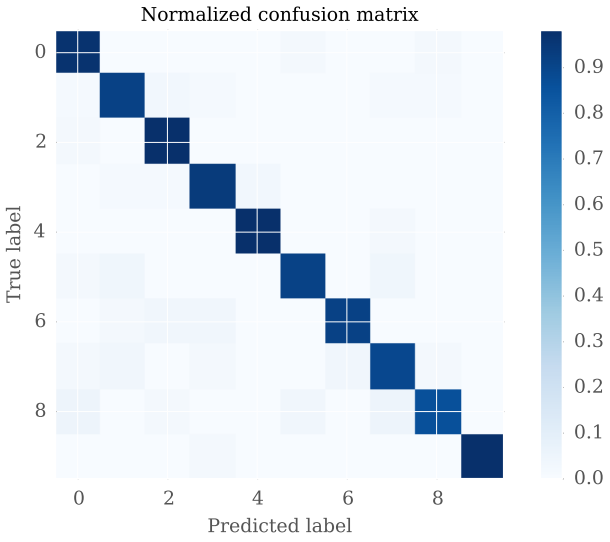

Figure 12. Confusion matrix for 10 gestures data set.

5. Discussion

While the results of our application are overwhelming positive there are several potential drawbacks of our system. The first drawback is that since our system employs a user-in-the-loop approach for hand segmentation, our system is highly sensitive to initialization; i.e, a bad initialization for the color profiling can result in spurious results. Since our system essentially uses skin hue range for segmentation, the system may not work too well if the background falls in the same hue range as the user's skin. To alleviate the effect of segmentation noise on our system, we employ Kalman filtering which can only reduce the spurious tracking to a certain point. If the initialization and hand segmentation is good, then our system performs well even in background clutter since only the descriptors falling within the hand blob are considered for bag-of-words features.

Another drawback of using the bag-of-words features is that it doesn't account for spatial layout. So the system would not be able to tell the difference between two gestures with similar bag-of-words features but different spatial layouts. For instance, the system may not be able to tell the difference between a gesture with the index and middle finger up versus the middle finger and the ring finger.

Also, the system may not generalize well on individuals it is not trained for, and the lighting conditions during test time should ideally be similar to those used during training data collection. However, this drawback of generalization to individuals and lighting conditions other than that of training data can be overcome with the proposed reinforcement learning method during test time. With enough feedbacks from the user, the system is able to converge for any individual and lighting condition.

6. Conclusion

Hand gesture recognition is a very rich mode for human computer interaction. Our system does a fairly good job of recognizing distinct hand gestures of the user using a traditional webcam. This paper explains several techniques we used for robust bag-of-words feature extraction, supervised model training and hyper-parameter optimization, and reinforcement learning model used for improving our system during test time. Unlike traditional gesture recognition systems that use only SVM or other supervised learning models, our system allows the user to give feedback to the reinforcement learning model, which over time does improve the results of the system.

7. Future Scope

With the integration of depth sensors within new webcams, one of the future scopes for our system would be to use depth cues for hand segmentation which will eliminate the need of user-in-the-loop approach for color profiling. Using depth cues for foreground segmentation would be more robust to background clutter. We can also recover 3D structure of user's hand with depth cues. This 3D geometry can be used as additional features to the classifier and can serve as spatial verification for our bag-of-words based prediction.

Recognition of both-hand gestures can be another fairly simple extension of our current system. For this, we can track and extract bag-of-words features from both the hands of the user to predict the gesture.

Since we are working with static gestures, there is a strong temporal correlation between user gestures. To incorporate this correlation, we can use a hidden Markov model, with the bag-of-words features as the observations and the true gesture ID as the hidden state. The system can output the predicted gesture ID based on the maximum likelihood estimate.This can also be extended to recognize dynamic hand gestures where the temporal nature of the hand movements are encoded within the trained hidden Markov model.

Finally, our current system implements reinforcement learning for weight updates only for linear SVM kernel. However, in future, we can implement it for other SVM kernels (especially for the additive chi-squared kernel, since it works well for histogram features). We can also implement reinforcement learning weight update for Neural Network using the backpropagation algorithm, which can be a potential future scope for our system.

References

Bergstra, James; Yamins, Dan and Cox, David D. Hyperopt: A python library for optimizing the hyperparameters of machine learning algorithms. In *Proceedings of the 12th Python in Science Conference*, pp. 13–20, 2013.

Caruana, Rich and Niculescu-Mizil, Alexandru. An empirical comparison of supervised learning algorithms using different performance metrics. In *Proceedings of the 23rd International Conference on Machine learning (ICML)*, pp. 161–168, 2005.

Chen, Qing; Georganas, Nicolas D. and Petriu, E.M. Hand gesture recognition using haar-like features and a stochastic context-free grammar. *IEEE Transactions on Instrumentation and Measurement*, pp. 1562–1571, 2008.

Dardas, N.H. and Georganas, Nicolas D. Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques. *IEEE Transac-*

tions on Instrumentation and Measurement, pp. 3592–3607, 2011.

Jang, Kyung Hyun; Shin, Do Kyung and Moon, Young Shik. A new snake for hand tracking using textural information. In *International Conference on Convergence Information Technology, 2007*, pp. 273–278, 2007.

Juan, Luo and Gwun, Oubong. A comparison of sift, pca-sift and surf. *International Journal of Image Processing*, pp. 143–152, 2009.

Kaelbing, Leslie; Littman, Michael and Moore, Andrew. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 1996.

Khaled, Hazem; Sayed, Samir G.; Saad El Sayed M. and Ali, Hossam. Hand gesture recognition using modified 1$ and background subtraction algorithms. *Mathematical Problems in Engineering*. URL http://dx.doi.org/10.1155/2015/741068.

Lagoudakis, Michail G. and Parr, Ronald. Reinforcement learning as classification: Leveraging modern classifiers. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, pp. 424–431, 2003.

Wiering, M.A.; van Hasselt, H.; Pietersma A.-D. and Schomaker, L. Reinforcement learning algorithms for solving classification problems. In *IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning (ADPRL), 2011*, pp. 91–96, 2011.

Wigdor, Daniel and Denis, Wixon. *Brave NUI World: Designing Natural User Interfaces for Touch and Gesture*. 2011.

Wu, Ying and Huang, Thomas S. Vision-based gesture recognition: A review. In *Gesture-Based Communication in Human-Computer Interaction*, pp. 103–115. 1999. URL http://dx.doi.org/10.1007/3-540-46616-9_10.