A Real-time Hand Gesture Recognition System

Arun Ganesan
University of Michigan, Ann Arbor

Caoxie (Michael) Zhang
University of Michigan, Ann Arbor

Abstract

Hello World

1 Introduction

Natural user interface (NUI) is a new way for human to interactive with machines. Among numerous NUIs which include multi-touch screen, eye tracking and many others, hand gesture seems to be one promising candidate. In this paper, we design and evaluate a novel hand gesture recognition system to demonstrate that we are close to an actual production-level system. The reader should be noticed that we do not claim hand gesture is THE future user interface, and in fact there are some limitations for using hand gestures such as users may feel fatigues (in the movie Minority Report, Tom Cruise has to take breaks many time due to fatigue). Similar to many other new UI systems, we propose our system as one (interesting) way to interact with the computer. We do not claim that the system would replace mouse and keyboard and we leave the usability problem to future research as building hand gesture recognition system alone is quite challenging.

1.1 Design Goals

Our system is designed to maximize user experience. Moreover, our system differs from other existing systems in the following ways.

Just hands. The users do not need to attach any additional physical objects to use our systems. They just need to show up their hands. Many existing system such as [SixSense, MIT Minority Report, MIT color glove] require users to wear gloves or markers for the RGB camera to capture. We eliminate this since the user should not do anything more than showing their hands.

Real-time. Our system should run smoothly on a modern machine with a graphical card not just on high-end machines. The system should also recognize hand gesture at a high frame rate. Our desired frame rate 30Hz, although the current version has around 5Hz. In our design and implementation, one driving goal is to squeeze every milliseconds as possible.

No calibration. Our system should not require a new user to do anything to calibrate the system to be used to the user.

Robust and Accurate. Our system should have an accurate estimation of where the users' hands are and what gestures they use with low false positive. Moreover, the system should be insensitive to various background, user's location, camera position and other noise.

Arbitrary gestures. Our system should be able to easily incorporate new types of gestures that any developers would like to add. By training new gestures, the system can recognize arbitrary gestures, for example the American sign language.

1.2 Main Ideas

Our system would not be possible without the use of Microsoft Kinect for PC, which we are probably among the first to obtain it in February 2012. Kinect is a multi-purpose sensor including RGB camera, depth camera and audio sensor. The Kinect SDK has offered skeletal recognition, which is however far away from recognizing hand gesture. The SDK also provides raw pixels for the RGB image and depth image at a maximum frame rate

30Hz. We use the depth image for gesture recognition and both RGB and depth image for generating training samples. The depth image is the key factor that distinguishes our system from most existing systems that use RGB camera. The advantage of the depth image is that it offers an addition dimension, i.e. depth of each pixel that is not present in the RGB image. An illustrating example would be an object and its background has similar color but the depth of the two is drastically different.

Our system adopts a data-driven approach: machine learning as opposed to hand-crafted rule-based systems. The adoption of machine learning transfers the human intelligent efforts from design rules/algorithms to design informative features. With the features on labeled data, machine learning allows computers to learn the rule/algorithm automatically. The key advantages of using machine learning in our system are (1) easy to incorporate developer-defined gestures: developers just need to feed the system with the gesture images to be trained rather than deriving new algorithms (2) robust to various environments such as camera position, background, various size of the hands: developers just need to generate the gestures on various environments without worrying about anything more.

In a high-level overview, the system is separated into two parts: training and real-time prediction. Only the real-time prediction component is seen by the end users. In the training component, we use color gloves to generate massive labeled data of the depth image. Random forest is trained to achieve both real-time performance and high accuracy. In the real-time prediction component, each pixel in the depth image is predicted the type of the gesture using GPU and then the prediction outputs are pooled to propose the final position and type of a gesture. Notice that we do not use any temporal or kinetics information as the current simple design suffices for the hand gesture tracking. [Need to elaborate more?]

1.3 Contributions

We summarize our contributions as follows:

A system for real-time hand gesture recognition. We design and implement a complete real-time hand gesture recognition system based on machine learning. The system can be used as API for other applications to read the gesture in real-time.

An computational insight about random forest and support vector machine (SVM). To the best of our knowledge, there seems to be no literature in comparing SVM and random forest from a computational perspective. We provide an in-depth comparison in the angle of performance rather than merely predictability as done in most machine learning literature.

Extensive experimental evaluations of the system. We conduct extensive experiments evaluation on the effectiveness of the machine learning approach, i.e., random forest we use in a large space of parameters. Interesting results reveals a deeper understanding of random forest.

1.4 Related Work [TODO]

There are two main techniques in hand gesture recognition - appearance based and model based. These are akin to probabilistic models of classification and generative models of classification respectively. **Appearance based** approaches read the pixels from the camera and build classifiers to label that as belonging to a finite set of classes. The main limitation of this technique is that the set of labels is finite and fixed ahead of time. The advantage of appearance based approaches is their implementation is often extremely fast and therefore suited for situations where live classification is important. Examples of appearance based techniques can be found in [1, 2]. **Model based** approaches start with a set of hypotheses of the final classification based on rules of the object being classified. For instance, in the case of hand gestures a hypothesis can be a particular orientation of the joints. An advantage here is that the hypotheses can be generated from an infinite space of possible classifications. The main disadvantage of model-based techniques is that they are often computationally expensive. In addition, model-based approaches tend to be very complex. An example of a model based technique can be found in [3].

2 System Architecture

There are three main stages to our system. First is acquiring training samples using a colored glove. Second is training a classifier on the samples. Third is live prediction and pooling.

The approach is pixel-level classification, and future pooling. The motivation for pixel-level classification is the potential of parallelizing the task and using the GPU to achieve real-time prediction.

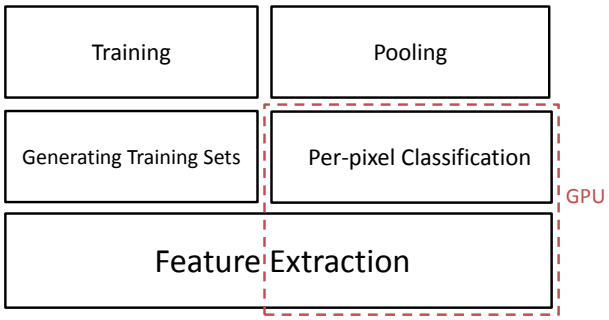

Figure 1: System architecture

3 Per-pixel Classification

3.1 Acquiring training samples

Color glove approach. To simplify the data acquisition step, we used cropping, depth thresholding, and a single colored glove. We used 4 gestures, giving a total of 5 classes including the background. We collected 400 training samples evenly spread across the gestures.

3.2 Training classifier

We trained the classifier on different subsets of the 400 training samples to study the effect of the training sample size. Also, we set aside 50 samples for testing purposes. Because our prediction is at the pixel level, we sample 1000 pixels from each of the training image, trying to get a good balance between the background and the gesture.

We used a depth-invariant feature vector for each pixel in our training sample. Each feature in the feature vector is the difference in the depth value of two offset pixels from the current pixel. We experimented with different number of such offset pairs for the feature vector.

We trained two classifiers - an SVM classifier for baseline testing, and a random forest classifier with multiple configurations. The random forest classifier can have multiple trees. We explored the effect of different number of trees in the forest.

3.3 Prediction

We aimed for real-time prediction of the hand gesture. The trained model from the classifier is used for per-pixel classification. That is, each pixel is classified as belonging to one of the four gestures or to the background. From here we have to first identify the likely gesture, and then find the location of that gesture, a process we called 'pooling'.

3.3.1 GPU

Running the prediction algorithm using the CPU proved to be very slow. For a 640×480 image, our system took 2.5 minutes to classify all the pixels. Given that this problem is highly parallelizable, we turned to the GPU. Re-implementing the prediction algorithm with the GPU reduced the prediction time from 2.5 minutes to 400 milliseconds - a 99.7% speedup!

3.4 GPU performance enhancements

4 Pooling

- mean versus median – mean is prone to outliers messing up the location. median is more resistant to that.
- majority gesture – if the hand is far away, this leads to the noisy labels being
- k-meniods – can be used to support multiple hands at once. and can be used to filter out the noise.

5 Implementation

Hello

6 Experimental results

6.1 Classification algorithm experiments

7 Experience

7.1 Limitations

Our current system is not without limitations. First, the real-time prediction component cannot achieve a frame rate at 30Hz but at about 5Hz.

8 Conclusion

References

[1] J. Shotton, A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman, A. Blake. Real-time human pose recognition in parts from single depth images. CVPR, 2011.

[2] R. Wang and J. Popović. Real-time hand-tracking with a color glove. In Proc. ACM SIGGRAPH, 2009.

[3] I. Oikonomidis, N. Kyriazis, and A. Argyros. Efficient model-based 3D tracking of hand articulations using kinect. In BMVC, Aug 2011.

[4] V. Lepetit, P. Lagger, and P. Fua. Randomized trees for real-time keypoint recognition. In Proc. CVPR, pages 2:775-781, 2005.

[5] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification Journal of Machine Learning Research 9(2008), 1871-1874.

[6] B. Schneiderman. The eyes have it: a task by data type taxonomy for information visualizations. Visual Languages, 1996.

[7] D.A. Keim. Information visualization and visual data mining. Visualization and Computer Graphics, IEEE Transactions. Vol 8, no.1, pp. 1-8, Jan 2002.

[8] W. Stuerzlinger, C Wingrave. The value of constraints for 3D user interfaces. Dagstuhl Seminar on VR, 2010.

[9] M. Hoffman, P. Varcholik, and J. LaViola. Breaking the status quo: improving 3D gesture recognition with spatially convenient input devices. IEEE VR, 2010.

[10] V. Bystritsky. ALGLIB. 14 Aug 1999. Web. `http://www.alglib.net`.