Identifying Hand Configurations with Low Resolution Depth Sensor Data

Sameer Shariff and Ashish Kulkarni

December 11, 2009

Motivation

In recent years, there have been many advances in the technology used to interact with computers, video game consoles, etc. including touch screens and input through use of accelerometers (Nintendo Wii). An natural extension to this trend is to enable human computer interaction though simple hand gestures and body movements. We set out to tackle a sub-problem of this task, detecting and classifying different hand configurations from low resolution depth sensor data. Naturally, with a good a hand configuration classifier, sequences of these configurations can be detected to identify full hand gestures. Applications of such a technology are widespread, including automatic sign language interpretation and computer and video game interaction.

Data Source

As our source of data, we're using the SwissRanger 4000, the 4th generation of series of Time-of-Flight cameras put out by Mesa Imaging. The camera uses infared light and measures phase differences in order to get an accurate measure of depth. Its output is a stream of frames, where each frames consists of a144x176 pixel image, and each pixel has an associated intensity *and* depth.

Problem

The problem we set out to solve was to detect and classifier different hand configurations from still frames of this data source. To simplify the problem, we focused on classifying two hand poses -- "opened" and "closed" hands -- but we allowed for a lot of variations in terms of the distance from the sensor, the orientation, and the angle from which we viewed the hand. The goal was to build a relatively robust classifier to differentiate between opened and closed hands in a variety of environments.

The End-to-End System

In the end-to-end system, we envision there being three major components:
1) Localize the hand (or hands) in the image, identifying the "sub-image" containing the hand
2) Recognize the configuration of the hand in the sub-image
3) Use sequences of hand configurations over time to recognize and classify full hand gestures

As mentioned above, for this project, we primarily focused on 2), manually labeling hand locations to feed into the recognition algorithm.

Data set

Our data set consisted of 162 still frames taken from our sensor, with the hand appearing at varying distances and orientations. All images contained a hand in either a closed or opened configuration.

Data Pre-processing

In a pre-processing step, we chose the region of interest from the full image based on a location which has manually input and a window size which was chosen based on the depth in the region around the hand. During this stage, we also labeled the hand configurations. All the data outside the region of interest was discarded, and all the data within the region of interest, including the intensity and depth at each pixel, was passed onto the next stage.

Feature Extraction

There were several types of features we developed, a few of which are described below.

2d shape-based features

These features typically used the depth information primarily to do segmentation, and then looked at the resulting contours in 2d space. In a few cases, we computed a mask binary image indicating which pixels fell within a certain bounding box of in the z-coordinate of the hand, and then used this mask image to derive features. Two of our more valuable features that came out of this process are described in more below.

Size-based features

One advantage of using a depth sensor is that we know an absolute measure of the scale of objects, since distances are no longer ambiguous (as they would be for a regular image). One idea was to use this as feature for classification, leveraging properties such as an open hand having more surface area exposed than a closed hand. Although these size-based features did add some value to our model, we found that in many cases, if the arm leading up to the hand was at a similar depth to the rest of the hand, this would get included in our segmentation and would shift this surface area measurement substantially. From looking through some of our failure cases, we found that these size based features were often too sensitive to what exactly was included in our region of interest.

Frequency-based features

Another observation we had is that an open hand (with fingers spread) has much higher frequency content than a closed hand. We developed a few different features to capture this effect. First, we computed the 2d discrete Fourier transforms of our images and used the Fourier transform weights as features. On their own, these features did add signal towards classifying open vs. closed hands, but we found that adding these features in addition to the 2d-shape based features described above added only a marginal gain. From this we concluded that our 2d shaped based features were, in effect, capturing some of this frequency content indirectly.

Specific Shape Features

The starting point for feature extraction for these features was the images depicted in sections (d) and (e) of Figure 1 below, after segmenting based on depth info and removing some of the noise in the image. Some of our priorities in our search for features that were scale and rotation invariant, and also resistant to small changes in the location and shape of the hand. We suspected that this would make it easier to properly detect open and closed hand configurations in different orientations. Below we describe two major features we have used so far which meet the above criteria.

Centroid distance histogram

We first computed the centroid of the white pixels, and then collected the distance between the centroid and each of the edge pixel in the image. To make this process scale invariant, the distance was normalized by diving each distance with the max distance. These distances were then bucketed and used as input features. Note that this process is invariant to scale and rotational changes.

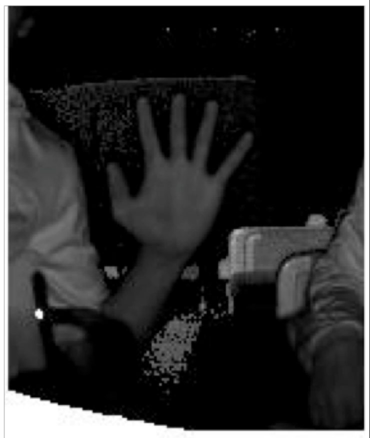
Edge pixel max distance histogram

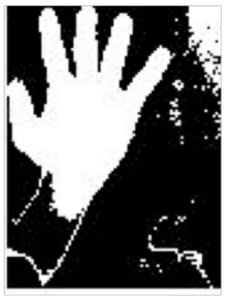
Similarly, this feature calculated the distance of an edge pixel with each other edge pixel in the image, and took the max across all these distances. This information captures some aspects of the shape of the hand. The resulting information about each pixel was again normalized using the max distance, and bucketed as our feature values. Similarly, this feature is also scale and rotationally invariant.

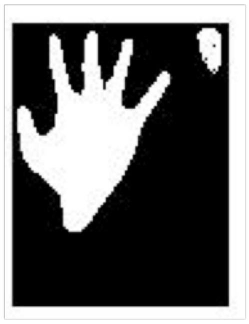
Learning algorithms tested

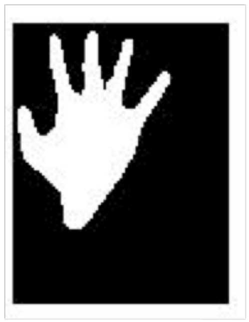
Although we tried a few different learning algorithms, we seemed to be getting the best results with SVMs so we focused our energy primarily on them. We trained our SVM using a linear kernel.

Figure 1 - Mask Images


(a) Original Image


(b) After using depth information


(c) After running density mask


(d) Remove small blobs

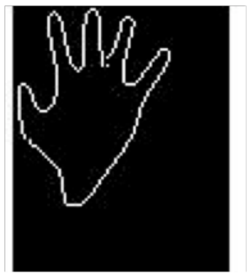
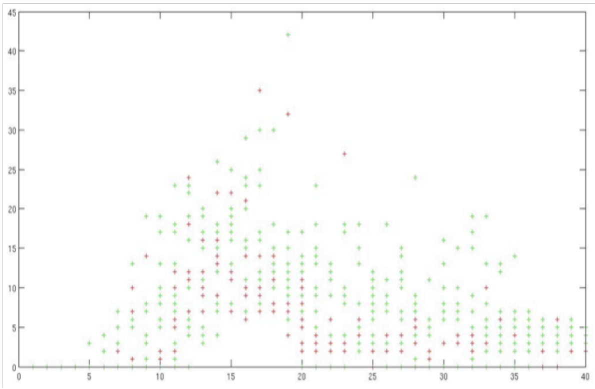
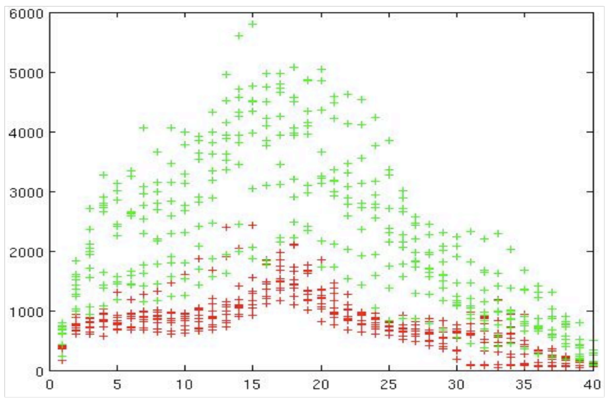

(e) After canny edge detection

Figure 2 - Histograms



Feature 1



Feature 2

Evaluation

To evaluate each of our models, we used K-fold cross validation with K = 10. We computed the test error for each of these 10 trials, and then computed the overall test error and test error variance as the mean and variance of these values, respectively. We chose our features and our parameters to maximize this overall test error.

Results

The table below summarizes our results. We've included the test accuracy of our model as each different feature type was added. That is, the test accuracy in the first row indicates that only discrete fourier transform feature was used. The second row indicates that the first two features were used, etc..

Feature Added	**Test Accuracy**	**Variance of Test Accuracy**
Discrete Fourier Transform	0.6294	0.0130
Edge distance of Each pixel from other pixels	0.8574	0.0045
Distance of Centroid from Edge Pixels	0.9010	0.0089
Measuring size of hand	0.9254	0.0068

Error Analysis

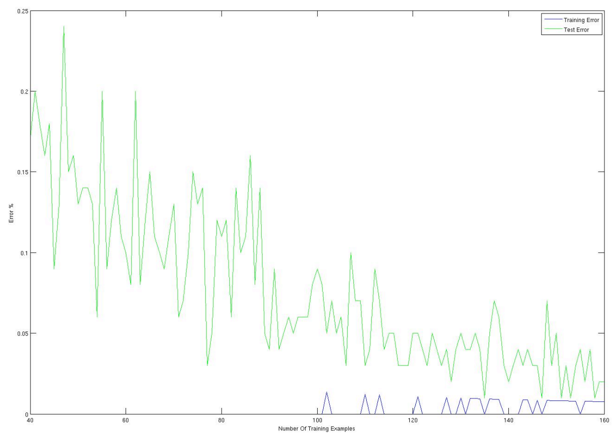
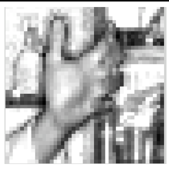
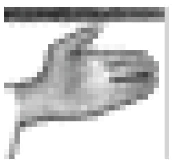
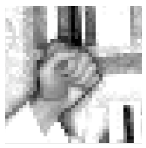
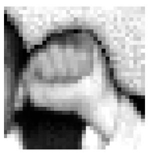
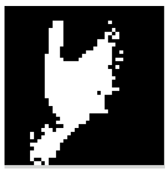
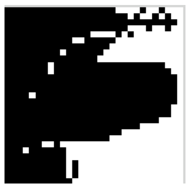
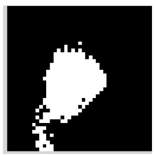
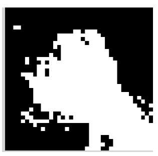


Figure 3 -
The graph on the right is a plot of the training and the test error, as the number of training examples is increased. The green line indicates the test error, and the blue line indicates the training error. The graph starts from plotting the training error when 40 examples were taken for training, and ends when 160 examples were taken for training. It can be seen that the test error decreases as the number of examples used for training increases.

A few examples of images which we classified incorrectly are included below. **Figure 4**

Wrong Mask Detection	**Error due to wrong depth estimation**	**Noise outside the hand**	**Features couldn't differentiate**
			
			

We analyzed the data which we got incorrect, and classified the reason of error into four parts. The four parts are
1) Wrong Mask Detection -- This is the case when there is error in the way mask is calculated. As it can be seen, in this image, the fingers got eliminated from the open hand during mask generation, and thus, this image looks more like a closed hand to the SVM classifier.
2) Error due to wrong depth Estimation -- This is the case when the sensor gave incorrect information about the depth in a particular region. This lead to classification of the hand as the background, and the background as the hand. As a result, the classifier was trying to classify incorrect data.
3) Noise outside hand -- This type of image image might be resulting due to one of out strong features. One of the features calculates the distance of each edge pixel from other edge pixels. The seems to be a lot of noise below the hand. Moreover, a part of the wrist is not detected, and thus giving it a weird shape below the hand. This results in the wrong classification by one of the features, which might be making the classifier have a wrong output.
4) SVM failing due to more than one of the above problems -- This last error type is when the classification fails due to multiple reasons. For example, this image has noise around it, after noise reduction, and there are holes in the middle of the wrist, which might mislead the SVM classifier.

Conclusions and Future Work

Our analysis above has shown that with a few carefully chosen features, we can derive a model of high accuracy to predict between two different hand configurations. The majority of our features above used an image representation as a starting point (after using depth data for segmentation) but several other features can be derived from the 3d representation of the data directly -- and this is an area we'd like to further explore. There are also several other directions we'd like to take this in terms of future work. First, naturally we'd like to extend this work to classify a larger number of hand configurations. Additionally, although we built some of our features into a real-time demo in C++, most of our features above are only implemented in Matlab, and we'd like to move these features over so we can do more accurate classification real-time. Finally, we'd like to work more on the hand localization tasks and gesture recognition in order to develop the full end-to-end real-time system.

Acknowledgements

We'd like to acknowledge Varun Ganapathi and Christian Plagemann for their guidance and input throughout this project. Varun and Christian provided the initial codebase through which we collected and began to visualize data from the sensor, and on many occasions they provided valuable feedback and helpful ideas on how to proceed. Varun and Christian have also provided several suggestions that we plan to plan to pursue beyond this project in order to build a more reliable and robust model.

References

[1] Andrew Ng. Support Vector Machines. CS229 Class Notes, 2009.
[2] Andrew Ng. Advice on Applying Machine Learning. CS229 Class Notes, 2009
[3] Canny, J. (1986), "A Computational Approach To Edge Detection", *IEEE Trans. Pattern Analysis and Machine Intelligence* **8**: 679–714, doi:10.1109/TPAMI.1986.4767851