Incorporating Gyroscope Information in Gesture Recognition

John Hiesey
jhiesey@cs.stanford.edu

December 16, 2011

1 Introduction

We expand and improve upon the Hidden Markov Model algorithm (KHMM) presented in Klingmann [1] to improve identification of 3d gestures on mobile devices. Specifically, we extend the algorithm to include a a weighted probability measure for clustering "fit", as well as expand our feature space to use gyroscope information, which is available on most mobile devices today. This gives nontrivial gains in both robustness and accuracy in 3d gesture recognition over Klingmann's initial implementation.

We alter KHMM's clustering assumptions. Specifically, we address the implicit assumption that clustering centroids remain uniform across gesture types. By assigning to each gesture type a unique set of centroids we make significant and immediate gains. We derive two alternative algorithms to test our hypotheses, denoting them Cluster-Matching-HMM and Cluster-Matching.

There are a variety of reasons why we use gyroscope information. Firstly, gyroscope data is additional information that is useful for identifying gestures, especially considering that slight variations in the device's orientation from gesture to gesture will affect accelerometer readings. These variations, however, can be corrected for with judicious use of gyroscope data. We hypothesize that a machine learning algorithm will be capable of utilizing the data to effectively make these corrections, thereby resulting in better orientation invariance. Secondly, if we wish to use 3d gestures as an input modality for mobile devices, inclusion of gyroscope information offers opportunities for greater expressiveness. Gestures can now include and be differentiated by orientation changes.

On a pragmatic note, we develop and test on the iPhone platform, although our solutions have general applicability, and an available Matlab implementation. The iPhone is chosen amongst our team as two of our members have previous iPhone development experience.

2 Algorithms

We now discuss the three main algorithms that we examine and analyze, starting first with Klingmann's baseline KHMM, followed by our Cluster-Matching-HMM and Cluster-Matching alterations.

2.1 KHMM

KHMM uses k-means clustering to discretize the space of real vectors followed by training one Hidden Markov Model for each gesture type. The intuition behind this can be thought of as follows: the clustering across all gesture types serves mainly to project each data vector which is in $\mathbb{R}^n$ (where there are n features) to $\mathbb{Z}$. Concretely this discretizes the data uniformly regardless of what gesture type it belongs to. Following the discretization KHMM then trains a Hidden Markov Model of the discretized training sets for each gesture type.

Formally we describe the algorithm as follows:

2.1.1 Clustering

For all training example across all gesture types, kmeans defines a clustering function $f : \mathbb{R}^n \to \mathbb{Z}$.

2.1.2 Hidden Markov Model

For each set of training examples for each gesture type train a Hidden Markov Model. Namely in this stage, for each gesture type g, we generate a function $h_g : (\mathbb{Z}^k) \to \mathbb{R}_{[0,1]}$ by the Hidden Markov Model process: $HMM : (\mathbb{Z}^k)^t \to h_g$, where each training example consists of k instantaneous data points and there are t training examples for gesture type g.

2.1.3 Classification Procedure

To classify an unknown gesture example, which exists in $(\mathbb{R}^n)^k$, we first transform it to its discretized version, namely $f(\mathbb{R}^n)^k$ and then for each gesture type g, apply $h_g((f(\mathbb{R}^n))^k)$ and classify the example according to the g were h_g returns maximum value.

2.2 Cluster-Matching-HMM

The motivation for Cluster-Matching-HMM derives from the realization that in the clustering step KHMM applies the same centroids to each training example regardless of what gesture type it belongs too. We hypothesize that different gesture types have vastly different vector features score (ie lie in different subspaces of $\mathbb{R}^n$) and thus we might gain information by clustering each gesture type independently and then training an HMM for each clustered training type. In the classification step we then classify over each clustering and each corresponding HMM. Moreover we weight the resulting HMM probabilities according to how well the training example "fits" in the the assigned cluster.

Formally we describe the algorithm as follows:

2.2.1 Clustering

For each gesture type g across all training examples, kmeans defines a clustering function $f_g : \mathbb{R}^n \to \mathbb{Z}$.

2.2.2 Hidden Markov Model

For each set of training examples for each gesture type train a Hidden Markov Model. Namely in this set we generate a function $h_g : (\mathbb{Z}^k) \to \mathbb{R}_{[0,1]}$ by the Hidden Markov Model process: $HMM : (\mathbb{Z}^k)^t \to h_g$, where each training example consists of k instantaneous data points and there are t training examples for gesture type g.

2.2.3 Classification Procedure

To classify a new example, which exists in $(\mathbb{R}^n)^k$, we first transform, for each gesture type g, the training example to its discretized version, namely $f_g(\mathbb{R}^n)^k$ and then apply $h_g((f_g(\mathbb{R}^n))^k)$. Finally, we classify the example according to the g' where $h_{g'} * p((f_{g'})^k)$ returns the maximum value. Here p is a "fittness value" of how well the training example fit into the given clustering. Namely we used an aggregate sum of the distances of the training vectors to their respective assigned centroids. Thus not only does each gesture type receive its own HMM but also its own clustering.

2.3 Cluster-Matching

Motivated by the effectiveness[1] of Cluster-Matching-HMM we define Cluster-Matching to naively cluster the training examples for specific gesture types. Then, when classifying, we assign an unknown gesture type to the gesture clustering to which it best fits.

Formally we describe the algorithm as follows:

2.3.1 Clustering

For each gesture type g across all training examples, kmeans defines a clustering function $f_g : \mathbb{R}^n \to \mathbb{Z}$.

2.3.2 Classification Procedure

For all g, compute $p((f_g(\mathbb{R}^n))^k)$. Return g that returns minimum value.

[1]See results.

3 Results

Clustering approach Our initial approach of clustering on all training data together worked reasonably well. When running on all 5 sample gestures and with the HMM enabled, we got an an accuracy of 89.6%.

However, on the same data, our cluster-matching algorithm combined with an HMM reaches 94.6%. We also observed similar improvements with examples, as seen in the data.

HMM Although we found a few examples where the cluster-matching algorithm benefits from using a Hidden Markov Model, in most real-world examples there is little benefit. When running on all of the training data, we found that adding the HMM actually decreased accuracy slightly, from 96.7% to 94.6%, which may be due to random fluctuations.

However, when comparing the data for circles to our contrived data set consisting of the first and second halves of each circle switched, the using an HMM with alphabet size of 20 improved the results from chance to 88.8%. Unfortunately we could not get such dramatic improvements with any non contrived data.

Normalization	Features	Algorithm	Gestures Types	Accuracy
Yes	Both	All	Cluster-matching with HMM (10)	94.6%
Yes	Both	All	Original	89.6%
Yes	Accel	All	Cluster-matching with HMM (10)	92.0%
Yes	Accel	All	Original	84.2%
Yes	Gyro	All	Cluster-matching with HMM (10)	92.3%
Yes	Gyro	All	Original	89.7%
Yes	Both	All	Cluster-matching (10)	96.7%
Yes	Both	All	Cluster-matching (20)	97.9%
Yes	Both	All	Cluster-matching (40)	98.4%
No	Both	All	Cluster-matching (40)	98.4%
Yes	Both	Squares	Cluster-matching (10)	100%
Yes	Both	Circles and Reversed-circles	Cluster-matching (10)	Chance
Yes	Both	Circles and Reversed-circles	Cluster-matching with HMM (10)	72.0%
Yes	Both	Circles and Reversed-circles	Cluster-matching with HMM (20)	88.8%

4 Conclusions

We worked hard, and achieved very little.

5 Future Research

Try our new algorithm with single clustering