# Programming-By-Example Gesture Recognition

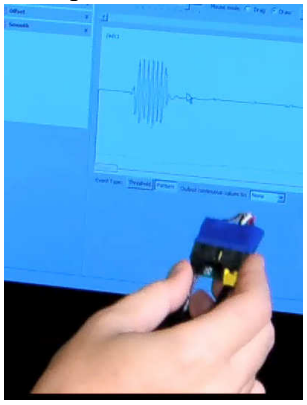
Kevin Gabayan, Steven Lansel

December 15, 2006

**Abstract**

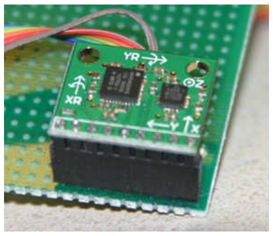
Machine learning and hardware improvements to a programming-by-example rapid prototyping system are proposed. Exemplar, a sensor interaction prototyping software and hardware environment, currently uses a dynamic time warping gesture recognition approach involving single signal channels. We use a five channel accelerometer and gyroscope combination board to sample translational and rotational accelerations, and a microcontroller to perform analog to digital conversion and relay incoming signals. Template matching via linear time warping (LTW) and dynamic time warping (DTW) are performed offline, as well as reinforcement learning via Hidden Markov Models (HMM) in real-time.

## Background

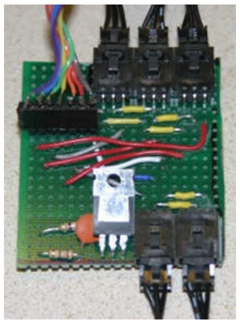


Exemplar is a current research project of the Stanford HCI group, a programming-by-example tool for interaction designers used to rapidly prototype interactions involving sensors. It consists of a microcontroller board, a variety of sensors, and a GUI used to visualize multiple signals, specify detection thresholds and windows, and map outputs. A user can perform a gesture using the available sensors and then view the gesture signal within a signal buffer. He can then markup the signal plot with signal threshold values beyond which an event will be triggered. The user may also select a time window that contains the gesture to train a gesture recognizer that will trigger events when the gesture is repeated. The pattern recognizer currently uses a dynamic time warping (DTW) algorithm that is intended to recognize gestures from a single training example, but it is not robust. We would like to improve the detection and classification accuracy of the gesture recognition system and allow a wider vocabulary of gestures to be detected by performing recognition across multiple signal channels. A demonstration video of the current system is available at http://hci.stanford.edu/exemplar/.

## Hardware



To increase the amount of information available to the gesture recognizer, we've used a five degree-of-freedom inertia measurement unit (IMU) combination board. The board includes the 3-axis Analog Devices ADXL330 accelerometer chip providing accelerations in X, Y, and Z axes and the dual-axis Invensense IDG300 gyroscope chip providing roll and pitch.



We've built a power and data relay board, which regulates USB power from 5V to 3.3V and relays the IMU board's analog outputs to the microcontroller ports. A long ribbon cable connects the IMU board to its power and data board so that users will only have to hold the small and light IMU board while performing gestures.

**Recording Template Gestures**

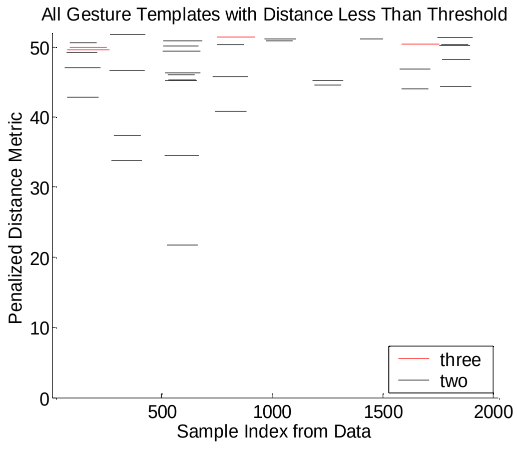
The IMU board output was digitally sampled at 100Hz at 10 bit resolution while sets of six to ten examples of each training gesture were performed while holding the IMU. These gestures included motions common to a kitchen, {Fry, Pound, TossPlate, RollDough, etc.}, motions common to sports/action games, {Lasso, Punch, TennisServe, Spy}, drawing numbers 1 through 4 in the air, and other miscellaneous gestures {WaveHello, ConductBeats, ChickenDance}.

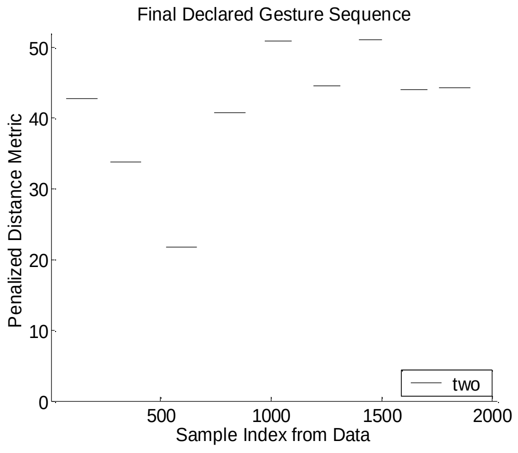
**Detection Algorithm using Linear Time Warping**

1. Smooth incoming data and all training gestures with rectangular window of length 20.
2. For each gesture in the training set, generate 9 gesture templates of varying lengths uniformly distributed over 50% to 150% of the original gesture length. The value for each sample of the scaled gesture templates was found using weighted averaging of the two sample values from the training gesture that are closest to the same percent from the beginning of the signal to the end as the original point in the template.
3. For each of the above gesture templates, select windows throughout the input data that have the same length as the template and use hop size of 5 samples.
4. The distance between each of the gesture templates and the window from the input data is calculated using an average absolute error metric for all three accelerometer readings
5. If the input data window has a variance lower than a particular predefined constant, we add a penalty to the above distance measure. The penalty is directly proportional to the deviation of the window variance from the predefined constant. The proportionality constant is chosen so that a window with variance 0 has a penalty equal to a predefined parameter, $\lambda$.
6. Any of the gesture templates that generate a penalized distance larger than a threshold are discarded. The remaining gesture templates are plotted in the figure below on the left.
7. Out of the remaining gesture templates, declare that the gesture template with the minimum penalized distance metric occurred.
8. Remove all remaining gesture templates that have a nonempty overlap in time with the recently declared gesture template.
9. Repeat steps 7 and 8 until there are no remaining gesture templates. The final result is plotted in the figure below on the right.

<u>Matching Algorithm Plots for a Series of Nine Consecutive ‘Number 2’ Gestures</u>

(Notice the ‘Number 3’ gesture is very similar to the ‘Number 2’ gesture so this is why the 3 gesture was a close match on the left.)





## Discussion of Algorithm Steps

Step 1: Smoothing is needed to remove noise in the accelerometer data and generate velocity-like signals that are easier to match. The length of the filter was chosen to adequately remove the noise but not destroy the underlying signal.

Step 2: We provide an example of how the scaling works. If we count the samples starting with 0, the value of sample 4 in a gesture template of length 101 samples generated from a training gesture of length 71 samples is equal to 0.2 times the value of sample 2 plus 0.8 times the value of sample 3 from the training gesture because 4*70/100=2.8.

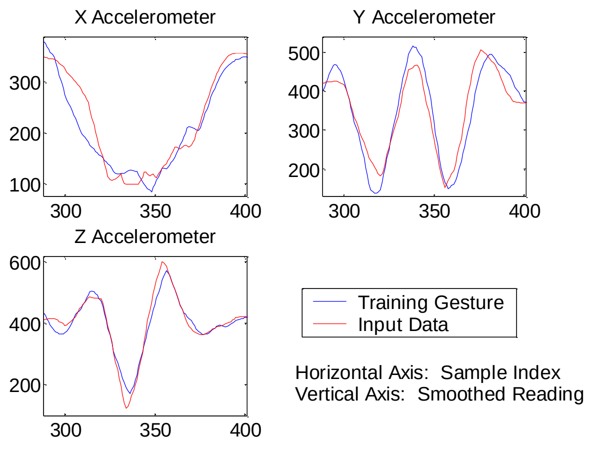
Step 5: Without the variance penalty, the algorithm generated a number of false positives. A majority of these errors occurred when the incoming data window had a small energy corresponding to periods of inactivity of the input device. In general, it is easy to match a gesture to a period of low activity since gestures in general have low energy. It was decided to use a variance penalty instead of a hard variance threshold so that gestures that match very well but have low variance can still be detected. Varying the parameter $\lambda$ adjusts the relative importance of the distance metric and variance penalty.

Step 8: Instead of discarding gesture templates that have nonempty overlap, it may be an improvement to only discard gesture templates that have an overlap larger than a small percentage of both gesture templates. This will make the algorithm more robust when the incoming data has a quick series of gestures without intermittent pauses.

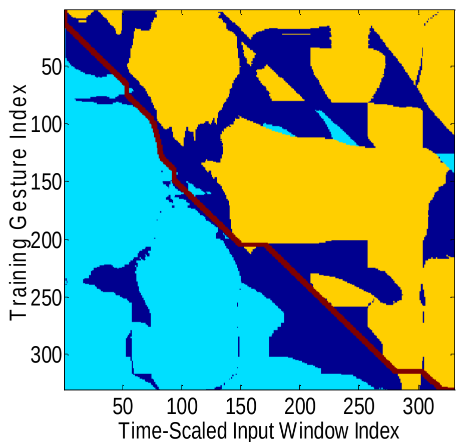
## Linear Time Warping Performance

The above algorithm has been tested in Matlab with 7 different gestures. A single instance of each gesture was used for training, and a total of 43 test gestures were processed. The algorithm successfully detected and classified all gestures and generated no false positives. The smoothed accelerometer readings for the input data and the correctly identified gesture are plotted in the figure below on the left.

Comparison of Input Data and Matched 'Number 2' Gesture



Dynamic Time Warping Optimal Matching



**Dynamic Time Warping (DTW)**

Although the above algorithm is able to account for gestures that are performed at a constant speed different than the training gesture using the windows of different lengths, it may not perform well for gestures that were performed at a number of different speeds relative to the training gesture. For example, the first part of a gesture may be performed slower than the training gesture and the second half may be performed very quickly. With linear time warping, which was implemented in the final algorithm, the two gestures will be misaligned in time. Dynamic time warping attempts to find the optimal alignment of the two gestures by solving a dynamic programming problem.

The DTW algorithm was implemented by changing only step 4 above. An optimal time alignment found with DTW is shown in the figure above on the right. The optimal alignment path is highlighted in dark red. For linear time warping, the only alignment path considered is the one that lies on the diagonal of the figure. Orange (light blue) pixels correspond to an optimal horizontal (vertical) movement from the pixel, and dark blue pixels correspond to a horizontal movement or a correct matching of those time instants. We decided the DTW was not needed for our current data set because our gestures had a relatively constant speed within each gesture; however, the DTW may be needed for more challenging data sets.

**Hidden Markov Models (HMMs)**

The gesture process that governs our IMU readings may be modeled with hidden Markov models (HMMs). This approach estimates the unobservable gesture process as a state at every time step and learns from training examples the probabilities of the process transitioning to other states in the model. Our implementation of this HMM classifier in Exemplar is based on the Georgia Tech Gesture Toolkit (GT2K), which implements the Cambridge University Hidden Markov Model Toolkit (HTK).

A motionless accelerometer produces a range of output values depending on its tilt due to gravitational acceleration, so the same gesture performed using multiple orientations will have a range of raw data values. To correct for this in individual dimensions, baseline values for each sensor are updated every time the accelerometer becomes still:

$$b_t := (1-\alpha)b_{t-1} + (\alpha)s_t$$

Where b is the baseline value and alpha is an update weight given to the new motionless sensor value, chosen to avoid abrupt changes in the baseline value and the affected energy measurement. The squared difference between sensor input and sensor baseline values is integrated over an analysis window to generate an energy measurement.

To assist in Exemplar's training and testing modes of data collection, gestures with energy above a threshold are automatically segmented. Doing so reduces the number of classifications performed by waiting for gestures of interest, but fails to pick out adjacent gestures that can be bounded by sliding windows. Without having a UI to allow label editing, after a training mode is begun each successive set of ten gestures is labeled with the same label. The gesture signal sequences are stored in a library of examples, and an eight-state HMM for each label present in the library is trained on its examples using the Viterbi algorithm. In recognition mode, a segmented gesture is labeled with the HMM whose transition matrix most likely produces an observation sequence best matching the input.

The system was trained to recognize digits drawn in the air with the IMU unit. Fifteen training examples of each digit drawn in the air with motionless periods separating each gesture were recorded and used to train the hidden Markov models. A test sequence of ten gestures of each digit produced the following confusion matrix:

| | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **1** | 0 | 1.0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **2** | 0 | 0 | 0.9 | 0 | 0 | 0 | 0 | 0 | 0.1 | 0 |
| **3** | 0 | 0 | 0.2 | 0.1 | 0 | 0.7 | 0 | 0 | 0 | 0 |
| **4** | 0 | 0 | 0 | 0 | 0.8 | 0 | 0 | 0 | 0.2 | 0 |
| **5** | 0 | 0 | 0 | 0 | 0 | 1.0 | 0 | 0 | 0 | 0 |
| **6** | 0 | 0 | 0 | 0 | 0.2 | 0 | 0.6 | 0.1 | 0.1 | 0 |
| **7** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.0 | 0 | 0 |
| **8** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1.0 | 0 |
| **9** | 0.2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0.8 |

**Confusion matrix.** The confusion matrix for the test set of 100 gestures. The digits in the horizontal entries are the correct labels of the input signals, and the digits in the vertical entries are the classifier outputs. The numbers represent the rate at which the inputs were classified under each target class.

These results represent an 82% digit recognition rate. Some digits were classified quite well, and some were classified perfectly. Digit three was more often confused as digit five. The similarities in the final strokes of both three and five may help explain for this confusion. The addition of more HMM states and data features may remedy an underfitting of data.

## Future Work

The classifiers may be further generalized by training upon gestures generated by multiple people, or generating multiple target classes that span the range gesture signals of many people performing the same gesture. Classifier parameters may be tuned for more optimal classification, such as those controlling sliding window sizes, window scaling, and HMM topologies. UI elements designed for entering training labels, viewing classifier outcomes, and removing poorly performed training examples would improve the Exemplar user experience.