# **Pattern Recognition Team Projects**

## **Overview**

There are several possible project topics in two categories:

* Domain Representation
  + Handwriting Vector Quantizer
  + Speech Input
  + Musical Instrument, Other?
* Recognition Engines
  + Dynamic Time Warp (Dynamic Programming Template Matching)
  + Hidden Markov Model
  + Nth-Order Markov Model
  + Neural Networks

The recognition engine teams will need test data so the Domain representation teams will need early results that the teams will be able to test and demonstrate their subsystem. A working recognition or synthesis system will require a subsystem from each category to be complete.

## **Team Membership**

Each team should be composed of students with a variety of backgrounds/skills so that the overall team can span the problem that their project will present. I expect that teams will each have 3 students.

## **Domain Representation Projects**

### **Goals**

* **Reversibility** – enhances application flexibility and testability
  + Flexibility comes from being able to use the subsystem in either recognition or synthesis mode.
  + Testability – Running a domain representation subsystem in synthesis mode (give it a representative feature sequence and output Speech/Writing) allows us to see/hear what the representation captures about the domain.
* Produce an **early result** (preferably mid-term) so that the Recognition Engine teams can have some good test data. The team can then enhance their subsystem performance as the semester continues.
* The **output format** should just be a sequence of named objects (e.g. integer digits 0-255) plus a developed closeness matrix that a recognition engine can use as a metric to help in “matching” dissimilar object sequences.

### **Handwriting Vector Quantizer**

The input to the subsystem is simply a sequence of {x,y,z} (z is either binary black/white or better yet pressure) coordinates from using a stylus to write on the screen. The output is a sequence of named vectors from a limited set (perhaps 16 angles and 8 lengths plus B/W). Things to consider:

* **Strokes** – Writing is a sequence of strokes. The stroke boundaries can be reliably determined by several methods (Newton, maxima/minima, and pressure) which allow the vector sequence within each stroke to be more consistent.
* **Normalization** – Character size, stylus speed and writer “Tilt/Slant” (very different for right handed vs left handed writers) all make the problem larger for the recognition engine. A Vector Quantizer should attempt to “normalize” these variables.
* **Closeness** – A straight forward “Euclidean Distance” metric can be used to produce a closeness matrix.
* **Word-Level Heuristic** – Some writing elements (dotting i’s, crossing t’s and Q’s) tend to be done at the word level instead of per character. This is a violation of our assumption of “smooth” over time. This can be detected by the Quantizer and a suitable triplet of strokes (2 white and one black) can be inserted into the stroke sequence of the appropriate letter. This enhancement will allow normal writing habits to continue. Of course the writer could get improved results by modifying his/her behavior to do these strokes in sequence at the letter level.
* **Reversibility** – Adding a smoothing algorithm to smooth out the quantized vectors when operating in synthesis mode would be a useful enhancement. Remembering the writer’s “tilt” and re-introducing it would also be useful in forging the writer’s handwriting.

### **Speech**

Classical speech recognition front ends eliminate phase and pitch information to reduce complexity stating that these do not add useful content for the recognition engine. The problem with that statement is that if you reverse the process, you or I would have difficulty recognizing the produced sounds so why are we surprised when the recognition engine has difficulty (especially with fricatives and plosives).

Speech consists of a sequence of sampled data from a microphone, which can be segmented into a sequence of “phones” which, in turn, make up a sequence of words (sometimes separated by silence). It is a more challenging problem than Handwriting and requires a much higher dimensional space for a good representation. In my earlier work (Smith, et al) we were presented with a speech front end that first passed the speech through a parallel filter bank (there were some anomalies in that design that caused difficulties), reprocessed the filter outputs into a “cepstral” vector space (note that all phase and pitch information was eliminated) and then used “Data Clustering” analysis to identify 256 most often occupied regions in this space and provide a closeness metric matrix for the set of regions (note that this set of regions tends to be speaker dependent). The sequence of names of these 256 regions (integer bytes) were then passed to the recognition engine along with the Closeness Matrix for the set.

## **Recognition Engine Projects**

### **Goals**

Each recognition engine subsystem needs to accept feature sequence objects and a distance metric matrix and determine the most likely state sequence that explains the observed feature object sequence. The actual meaning of the output state sequence is problem dependent:

* **Handwriting** – The output state sequence is hopefully an accurate alpha-numeric text sequence representing the intent of the author.
* **Speech** – The output state sequence can either be a sequence of “Phones” (as I did in my earlier research that then was used with a second engine to produce the alpha-numeric text sequence corresponding to the speakers voice input) or a sequence of words from a pre-defined set that were spoken into the microphone.

### **Dynamic Time warp (DTW)**

Here a set of templates representing input objects sequences that occurred in a sample set of repetitions (either averaged in some manner, or a subset of templates, one for each repetition) of each of the states in the problem is stored and the unknown sequence is locally stretched/compressed (penalized each time) to fit each template. The state that requires the least adjustment to get a match is declared the best fit.

DTW has been successfully used in both speech and handwritten recognition problem areas.

### **Hidden Markov Model (HMM)**

Here the input objects are treated as a set of “observables” that are generated during transitions between Markov states that are not directly observable. There is a separate HMM that is developed to represent each system state and the sequence of HMMs that best explain the observed objects gives the recognized application state sequence. [Do not be confused by the fact that there are application states (letters, phones, words, ….) and also HMM internal states]. HMM systems were developed at IBM and form the basis for most speech recognition systems today.

### **Nth Order Markov**

Here the Markov state is explicitly given by the most recent n-1 observed objects. The model is a variable order Markov data structure (an augmented tree) that was patented as a “Context Organized Memory” (COM) – see Smith et. al. (1985). This approach is computationally efficient and can be incrementally trained while in use.

### **Neural Networks**

A Neural Network based model assumes a multi-layered, fixed arrangement of elements who’s interconnection parameters are updated incrementally to produce that desired recognition functionality. This approach has a long development history dating back to the “Perceptrons” of the 1950’s.