Georgia Tech Gesture Toolkit: Supporting Experiments in Gesture Recognition

Tracy Westeyn, Helene Brashear, Amin Atrash, and Thad Starner
GVU, College of Computing, Georgia Institute of Technology
801 Atlantic Drive, Atlanta, Georgia
{turtle, brashear, amin, thad}@cc.gatech.edu

ABSTRACT

Gesture recognition is becoming a more common interaction tool in the fields of ubiquitous and wearable computing. Designing a system to perform gesture recognition, however, can be a cumbersome task. Hidden Markov models (HMMs), a pattern recognition technique commonly used in speech recognition, can be used for recognizing certain classes of gestures. Existing HMM toolkits for speech recognition can be adapted to perform gesture recognition, but doing so requires significant knowledge of the speech recognition literature and its relation to gesture recognition. This paper introduces the Georgia Tech Gesture Toolkit **GT^{2}k** which leverages Cambridge University's speech recognition toolkit, HTK, to provide tools that support gesture recognition research. **GT^{2}k** provides capabilities for training models and allows for both real–time and off-line recognition. This paper presents four ongoing projects that utilize the toolkit in a variety of domains.

Categories and Subject Descriptors

I.5 [**PATTERN RECOGNITION**]: I.5.mMiscellaneous; H.5 [**INFORMATION INTERFACES AND PRESENTATION**]: H.5.2User Interfaces; G.3 [**PROBABILITY AND STATISTICS**]: Markov processes

General Terms

Design, Experimentation, Human Factors

Keywords

Gesture Recognition, Interfaces, Toolkit, Hidden Markov Models, American Sign Language, Context Recognition, Wearable Computers

1. INTRODUCTION

The Georgia Tech Gesture Toolkit (**GT^{2}k**) provides a publicly available toolkit for developing gesture–based recognition systems. The toolkit allows easy development of the gesture recognition component of larger systems. Figure 1 shows the integration of **GT^{2}k** into such a system. First, sensors such as video cameras or accelerometers gather data about the gesture being performed. This sensor data can be processed to ascertain the salient characteristics, known as features. The *Data Generator* collects this data and provides the features that are used by **GT^{2}k** components to perform training and recognition. The results returned by **GT^{2}k** are considered by the *Results Interpreter* and acted upon based on the needs of the application.

Figure 1: GT^{2}k interaction with application components.

GT^{2}k allows researchers to focus on developing systems that use gesture recognition and the research surrounding those projects, instead of devoting time to recreate existing gesture recognition technology. **GT^{2}k** abstracts the lower level details of the pattern recognition process and allows users to focus instead on high level gesture recognition concepts by providing a suite of configurable tools. Appropriate applications for **GT^{2}k** are systems which utilize discrete gestures, such as sign language, handwriting, facial gestures, full body activities, and issuing robot commands. **GT^{2}k** is not designed for the creation of tracking devices such as those that might be used for controlling a mouse [4]. This toolkit may be of interest to researchers in the areas of human–computer interaction, assistive technologies, robotics, and other fields involving gesture recognition.

In this paper, we present the motivation behind the development of **GT^{2}k**, outline the functionality it provides, and demonstrate its use as a component of a larger system. To establish the flexibility of the **GT^{2}k**, we introduce four ongoing applications from different domains: The Gesture Panel, Prescott, Telesign, and Workshop Activity Recognition.

2. MOTIVATION

Hidden Markov models (HMMs) are probabilistic models used to represent non-deterministic processes in partially observable domains, and are defined over a set of states, transitions, and observations. Details of HMMs and the respective algorithms are beyond the scope of this paper but may be found in Rabiner's HMM tutorial [8].

ICMI'03, November 5–7, 2003, Vancouver, British Columbia, Canada.

The speech–recognition community has invested significant resources into development of recognition technology. The Hidden Markov Model Toolkit (HTK) [1, 11], an open source HMM toolkit, was developed for speech recognition applications. $\mathbf{GT^2k}$ serves as a bridge between the user and HTK services by abstracting away the lower level speech–specific functionality and allowing the user to leverage the full power of HTK's HMM manipulation tools. $\mathbf{GT^2k}$ allows the gesture–recognition community to benefit from the speech–recognition community's research by providing a tool powerful enough to satisfy the needs of people versed in HMM literature but simple enough to be used by novices with little or no experience with HMM techniques.

Research involving the recognition of American Sign Language (ASL) has shown that the same HMMs based techniques used for speech recognition can also be applied to gesture recognition [5, 10, 9]. Recognition of a limited ASL vocabulary from video has been demonstrated using HTK [9]. Adapting HTK from speech to gesture recognition required broad knowledge of the speech recognition literature and a deep understanding of HMMs. While HTK proved to be a very powerful tool, its application to a non-speech recognition domain was time consuming. $\mathbf{GT^2k}$ addresses this problem by providing a development tool specifically designed for gesture recognition.

3. USING $\mathrm{GT^2k}$

$\mathbf{GT^2k}$ provides a user with tools for preparation, training, validation, and recognition using HMMs for gesture–based applications. Preparation requires that the user design gesture models, determine an appropriate grammar, and provide labeled examples of the gestures to be trained. Training uses information from the preparation phase to train models of each gesture. Validation evaluates the potential performance of the overall system. Recognition uses the trained models to classify new data. At this point, $\mathbf{GT^2k}$ assumes data is being provided by a *Data Generator*, such as a camera, microphone, or accelerometer, in the form of a feature vector. The resulting $\mathbf{GT^2k}$ classification is then handled by a *Results Interpreter* as appropriate for the application.

We will ground our discussion of $\mathbf{GT^2k}$ components with a fictitious example involving simple military hand gestures for directing ground vehicles. The set of gestures is limited to *Attention*, *Halt*, *Advance*, *Reverse*, and *Slow_Down*.

In our example, a single soldier is providing directions for positioning of a ground vehicle. Before any gesture is made, the attention of the driver must be acquired. We will assume that the vehicle must always be instructed to slow down before halting. The vehicle can be repeatedly instructed to advance or reverse until properly positioned.

3.1 Preparation

The preparation phase provides $\mathbf{GT^2k}$ with set of initial gesture models, a semantic interpretation of the data, and examples of each gesture for training.

3.1.1 Designing Gesture Models

Each gesture is modeled using a separate HMM. Hidden Markov models are specified by a topology which includes a set of states and transitions. These transitions define the connections between the states and the likelihood of following that connection. HMM training can update the probabilities of transitions but does not typically add or remove states from the topology. Thus, designing the gesture models involves some insight into the structure of the data and may require some experimentation to determine the best topology. $\mathbf{GT^2k}$ provides tools allowing novice users to automatically generate models, while still providing experienced users with the capabilities to craft models which incorporate domain-specific knowledge. Visualization tools are also provided to aid in model construction. An example visualization of an HMM topology can be seen in Figure 7.

For our example, *Attention*, *Halt*, *Advance*, *Reverse*, and *Slow_Down* are represented by five distinct HMMs with identical initial topologies.

3.1.2 Specifying a Grammar

In the simplest case, recognition can be performed on one gesture at a time. This technique is known as *isolated* gesture recognition. However, sometimes it is necessary to perform *continuous* recognition on a sequence of gestures within a contiguous block of data. Knowledge of the possible sequences of gestures can be presented to $\mathbf{GT^2k}$ in the form of a rule–based or stochastic grammar. Grammars allow $\mathbf{GT^2k}$ to leverage knowledge about the structure of data, which aids in continuous recognition by using previously classified gestures to constrain the current gesture classification. Grammars also allow users to define complex gestures as a sequence of simpler gestures.

As stated in our example there is an enforced structure to the ordering of the gestures. This structure can be represented as a rule–based grammar. The following grammar uses "|" to denote a choice of gestures and "<>" to indicate one or more repetitions of the enclosed expressions.

```
MoveForward = Advance Slow_Down Halt
MoveBackward = Reverse Slow_Down Halt
command = Attention <MoveForward | MoveBackward>
```

3.1.3 Data Collection and Annotation

Sensing devices are used to gather data about activities in the environment. Common sensing devices include cameras, accelerometers, microphones, laser range finders, and motion capture devices. These sensors typically return raw data measurements of the observable environment. This raw data can be used directly for recognition or processed to extract the significant features as deemed appropriate for the task. This data is stored as numerical vectors, known as *feature vectors*, and form the data set over which $\mathbf{GT^2k}$ operates. The range of values and the length of the vector is dependent on the application. A typical gesture example would appear as a sequence of these feature vectors. For instance, video-tracked gestures may return the position of the hand at each video frame as an (x, y) coordinate. This results in a feature vector of length two with two real values corresponding to (x, y) position. A gesture which completes in 29 frames would be represented with 29 feature vectors.

In order for $\mathbf{GT^2k}$ to properly understand the data it receives, the data must be annotated by the user. This requires the user to specify which gestures appear in each of the training examples.

Using our example, assume a camera is tracking the hand of the soldier as he issues the following gesture sequence: *Attention Advance Slow_Down Halt*. The entire sequence is captured in 250 frames of video resulting in 250 feature vectors. Each feature vector consists of (x, y) position of the hand in the video. An annotation of this data may appear as follows:

```
1   56  Attention
57  175 Advance
176 235 Slow_Down
236 250 Halt
```

3.2 Training

The major contribution of $\mathbf{GT^2k}$ is the abstraction of the training process. Once the preparation phase is complete, training of the model requires only that the user select a training validation method and configure a few method–specific parameters. The training process is automated, returning results and models which can later be used for recognition in various systems. This abstraction allows users to avoid the details of the underlying algorithms. $\mathbf{GT^2k}$ provides a few default training/validation methods, however user–defined methods can easily be integrated.

Training/validation methods provide quantitative feedback concerning the training process. Such methods typically require that data collected for training be separated into two sets, a *training set* and a *validation set*. The training set is the set of data used to train the models, and the validation set is used to measure the performance of the trained models on unseen, yet known data. Evaluation of the model using the validation set helps gauge how well the model will generalize to new data. It also helps determine if overfitting occurs during the training process. Overfitting results in improved performance over the training data but a decline in generalization, and thus a decrease in performance over new data.

Two standard training/validation techniques provided by $\mathbf{GT^2k}$ are *cross–validation* and *leave–one–out validation*. Cross-validation randomly selects a predetermined percentage of the data (typically 66.6%) as the training set. The remaining data (typically 33.3%) acts as the validation set. Leave–one–out validation selects one data example as the validation set and uses the remainder of the data as the training set. The training/validation phase is repeated for every permutation of the data set with one element "left out" of the training set. The results of each iteration are then tallied to compute overall statistics of the models' performance.

Established HMM algorithms are used to train the models. Initial model parameters are estimated using Viterbi alignment. The parameters are then updated via several iterations of Baum–Welch re–estimation. This process generates models which represent the training set examples.

Returning to the example, assume a traffic director was observed and 100 sequences of his gestures were recorded. If cross–validation was used during training, 66 random examples would be selected to train the models, and the system performance would be determined based on recognition of the remaining 34 examples. Leave–one–out validation would select the first example, train on the remaining 99, and perform recognition on the removed example. This process would repeat for each of the 100 examples, i.e., the next iteration would remove the second example and train on examples 1 and examples 3 through 100.

3.3 Validation and System Performance

The training/validation methods provide a quantitative measure of the system's performance based on the accuracy of recognition. The $\mathbf{GT^2k}$ metric for accuracy is the standard definition that incorporates substitution, insertion, and deletion errors. Substitution errors occur when the system incorrectly classifies a gesture. Insertion errors occur when the system hallucinates the occurrence of a gesture. Deletion errors arise when the system fails to recognize the occurrence of a gesture within a sequence of gestures. If we let S represent substitutions errors, I represent insertion errors, D represent deletion errors, and N represent the total number of examples, then accuracy is defined as:

$$Accuracy = \frac{N - S - D - I}{N}.$$

It should be noted that insertion and deletion errors can only occur during continuous recognition. When recognizing gestures in isolation, the values for D and I will always equal zero.

System performance is reported in the form of a confusion matrix. The matrix reports the ground–truth gesture versus the gesture as classified by the system. An example confusion matrix can be seen in Figure 3.

3.4 Recognition Application

Models generated during the training phase can be used for recognition of new data. As with the training phase, the underlying algorithms have been abstracted. Once an unclassified gesture is received, $\mathbf{GT^2k}$ calculates the likelihood of each model. The actual probabilities are calculated using the Viterbi algorithm. This information can be used by the results interpreter as deemed appropriate by the application, for example, taking the most likely gesture or considering the probability of each gesture for making a decision.

Once a model of each of the gestures has been trained, it can be used independently of the training system. For our fictitious example of recognizing gestures for controlling ground vehicles, the individual models could be embedded in an autonomous robot. When the traffic controller gestures towards the robot, $\mathbf{GT^2k}$ receives features from the robot's sensors, calculates the probability of each model given the features, and returns a list of the most likely gestures issued by the traffic controller. The robot acts accordingly based on this data.

4. APPLICATIONS OF $\mathrm{GT^2k}$

To demonstrate the flexibility of $\mathbf{GT^2k}$, we will present four projects currently utilizing the toolkit: The Gesture Panel, Prescott, Telesign, and Workshop Activity Recognition.

4.1 The Gesture Panel: Gesture Recognition in the Automobile

Interior distractions are a large factor in automobile accidents in the United States. For example, changing the radio station while driving involves both taking a hand off the steering wheel and, more dangerously, glancing repeatedly at the radio to determine which button to press. We are developing a system called the Gesture Panel which allows the driver to control devices using simple, gross gestures with a minimal amount of distraction. Such a system allows the driver to keep his eyes on the road while controlling a device.

Varying lighting conditions pose a major problem for gesture recognition inside automobiles. For example, if a car passes through a tunnel on a sunny day, the interior of the car will change from high to low illumination as it enters the tunnel. Any vision system for an automobile must be able to cope with lighting changes of this type. To avoid

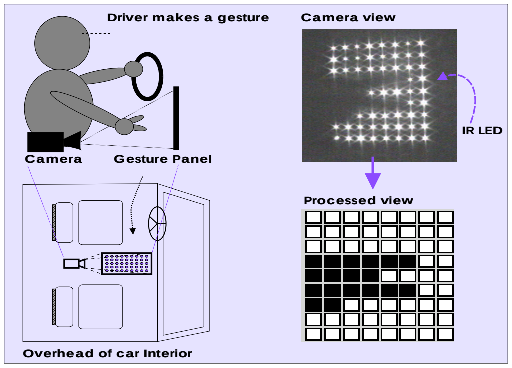

Figure 2: Gesture Panel in a vehicle. Right: Overhead and side view of Gesture Panel placement. Left: Camera view of gesture and corresponding binary representation.

this problem, the Gesture Panel employs a black and white camera pointed at a grid of infrared light emitting diodes (IR LEDs). Preliminary experiments with a prototype gesture panel demonstrate that the light from the IR LEDs is brighter to the camera than any reflected sunlight present in the car. The aperture of the camera is reduced so that only the light from the IR LEDs is visible. Gestures are performed between the camera and the grid as shown in Figure 2. As the gesture is made, light from some of the LEDs is occluded by the hand. Gestures can be recognized based on the various patterns of occlusions through time.

4.1.1 Preparation

Designing Gesture Models: The gestures performed by the driver should be as simple as possible so as not to distract the driver from the task of operating the car. For our preliminary research, a total of eight gross gestures were selected. The gestures consist of a hand sweep across the panel in one of eight directions: up, down, left, right, and the four diagonal directions. Through experimentation we discovered that we could model the gestures with an eight–state left–right HMM topology. The same initial parameters were used for each of the eight models.

Specifying a Grammar: For this specific experiment, only isolated gestures are identified and they can occur in any order. Therefore, the grammar used by $\mathbf{GT^2k}$ will have the form:

```
gesture =  up | down | left | right | upper-left |
upper-right | lower-left | lower--right
```

Data Collection and Annotation: The feature vector generated for $\mathbf{GT^2k}$ directly corresponds to the number of IR LEDs present in the grid. The current prototype of the Gesture Panel has an eight–by–nine array for a total of 72 IR LEDs. The feature vector, therefore, is a 72–element array of binary values. The binary values indicate whether or not the LED is occluded from the camera.

4.1.2 Training, Validation, and System Performance

Leave–out–one validation was selected to train and test the models. The system correctly classifies 249 of the 251 examples. This is an accuracy of 99.20%. The confusion matrix for the system is shown in Figure 3.

```
----------------- Overall Results ------------------
%Corr=99.20, Acc=99.20 [H=249, D=0, S=2, I=0, N=251]
----------------- Confusion Matrix -----------------
            d   u   l   r   u   u   d   d
            o   p   e   i   p   p   o   o
            w       f   g   l   r   w   w
            n       t   h   e   i   n   n
                        t   f   g   l   r
                            t   h   e   i
                                t   f   g
                                    t   h
                                        t
            -   -   -   -   -   -   -   -
down       25   0   0   0   0   0   0   0
 up         0  25   0   0   0   0   0   0
left        0   0  36   0   0   0   0   0
right       0   0   0  31   0   0   0   1
up_left     0   0   0   0  34   0   0   0
up_right    0   0   0   0   0  27   0   1
dwn_left    0   0   0   0   0   0  33   0
dwn_right   0   0   0   0   0   0   0  38
====================================================
```

Figure 3: Confusion matrix for the Gesture Panel. Rows indicate the actual data labels, while columns depict the system's classification of the data. Of the 251 examples, 249 were classified correctly, and 2 examples were incorrectly classified as down_right gestures.

4.1.3 Recognition Application

The models trained by $\mathbf{GT^2k}$ have been successfully integrated into a real-time recognition system. In the future, this prototype system will be used to study the types of gestures that are appropriate for use within an automobile.

4.2 Prescott: Patterned Blink Recognition

Restricted access is a necessary component for protecting sensitive areas within public buildings, such as airports, research labs, and government buildings. Imagine a secure area located in a major airport. To ensure that only authorized personnel have access to the area, a numerical keypad controls the locking mechanism on the door. To unlock the door the correct code must be entered on the keypad. Because access is based solely on entering the correct number, an unauthorized person can foil the system by observing the correct code and then entering it. A method to protect against this is to include biometrics. A camera can be placed in front of the door and access can be controlled based on face recognition and entering the correct personal identification number (PIN). However, this system is flawed, as the face recognition system can be fooled by placing a photograph of an authorized person in front of the camera.

To address this issue, one might replace the numeric keypad with a system that requires the person to blink a specific pattern. This system can utilize the hardware already in place for face recognition and help improve robustness. Several benefits could be introduced by such an augmentation. First, replacement of the keypad would allow hands–free entry through the door. Second, the rapid movement of the eyes during a blink can be used to localize the position of the head in the video [2], which can be beneficial to the face recognition portion of the system. The blinking can also reduce the probability of someone deceiving the face recognition by placing a photograph in front of the camera, because the face is now required to have a dynamic component. Third, a personal blink pattern may be more difficult for a third party to observe because the user's blinks can only be viewed by another person if the observer is watching the user at an angle similar to the camera. Since the user will be facing the camera to perform his blinks, he will be more likely to notice a person attempting to observe his code (in comparison to someone peering over his shoulder as he enters numbers into a keypad).

Prescott is a prototype system used to investigate if the intrinsic properties of how a person blinks a specific pattern, their "blinkprint" (a "blinking fingerprint"), can be used to perform identification. In this case, recognition would depend on more than just the pattern itself; it could also be dependent on the time between blinks, how long the eye is held closed at each blink, or other physical characteristics the eye undergoes while blinking. Figure 4 shows the visible difference between two people blinking the same pattern.

4.2.1 Preparation

Designing Gesture Models: A separate model is used to represent the blink pattern performed by each individual. In this specific experiment, there are three participants and thus three corresponding models. Since each participant is blinking the same pattern, all initial models will have the same topology and same initial parameters. Through experimentation we discovered that nine–state, left–right HMMs were sufficient for modeling each individual's blinkprint.

Specifying a Grammar: Participants can approach the system in any order. Therefore, the grammar used by $\mathbf{GT^2k}$ will have the form:

```
blinkprint =  Participant1 | Participant2 | Participant3
```

Data Collection and Annotation: The participants in our experiments were required to situate themselves in front of a camera and a video monitor. They then had to align themselves in such a way that the video image of their face matched an outline drawn over the video monitor. Once aligned, the participant blinked several repetitions of the pattern "— — — . . — — —" where '—' and '.' represent long and short blinks respectively. The duration of the blink can be both the amount of time the eyelid is held closed and/or the duration of time between each blink.

Optical flow [6] is used to extract features from the video and detect when a blink occurs. Using optical flow helps to provide robustness by producing features that remain consistent across varying environments. Because eye blinks are very rapid as compared to head movement, changes in the optical flow field over the eye allow accurate detection of changes of motion in the eyelid. The location of the eye is known in advance. This *a priori* knowledge allows us to search only the necessary portion of video (a 64x64 region over the left eye) for detection of blinks. This constraint allows the calculation of optical flow to remain practical. However, knowing the location of the eyes is not required; it has been shown that the motion of the eyelids is sufficient for detection of the eyes' location in video [2].

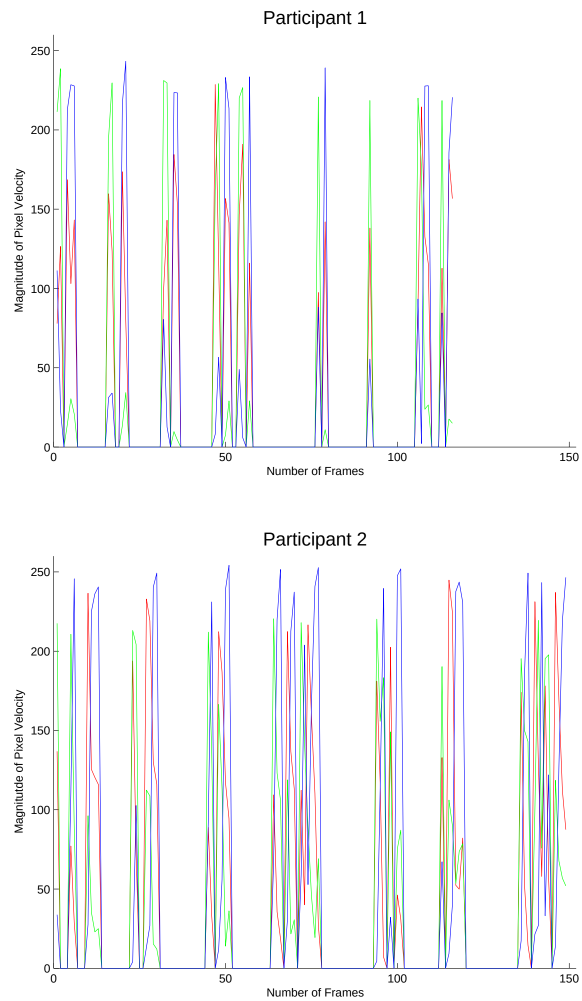

Figure 4: A visual representation of the same 9–blink pattern, "— — — . . — — . ." performed by two different people where '—' and '.' represent long and short blinks respectively. The duration of the blink can be measured by both the amount of time the eyelid is held closed and the duration of time between each blink. Visually the blinks are most easily discerned by the space between them. Often the space between the short blinks is difficult to locate through visual inspection. For example, the fourth peak of Participant2 (around frame 75) represents 2 short blinks.

The movement of a pixel is represented as a velocity vector expressing the magnitude and direction of change from a previous frame. Because we are looking for a motion that will be rapid when compared to the rest of the motion in the image, we ignore pixels with small velocities. This policy has the effect of filtering out natural motions, such as slight oscillations of the head position, while the sequence is performed. The mean velocity is computed over the salient pixels. This velocity vector can be represented as a three element vector supplied to $\mathbf{GT^2k}$.

4.2.2 Training, Validation, and System Performance

Leave–one–out validation was selected to train and test

```
------------------ Overall Results -----------------
%Corr=89.58, Acc=89.58 [H=43, D=0, S=5, I=0, N=48]
------------------ Confusion Matrix ----------------
                P   P   P
                1   2   3
                -   -   -
Participant1   14   0   0
Participant2    0  13   5
Participant3    0   0  16
====================================================
```

Figure 5: Confusion matrix for the Prescott. Rows indicate the actual data labels, while columns depict the system's classification of the data. Of the 48 examples, 43 were classified correctly, and 5 examples were incorrectly classified as "Participant3" blinkprints. Column classification labels abbreviated for clarity.

the models. The system correctly classifies 43 of the 48 examples. This is an accuracy of 89.6%. The confusion matrix for the system can be seen in Figure 5.

4.2.3 Recognition Application

Currently a real-time recognition system has not been implemented. In the future this system will be expanded to incorporate more participants using a variety of patterns and allow for real-time recognition using models generated by $\mathbf{GT^2k}$.

4.3 TeleSign: Mobile Sign Language Recognition

In previous work, we demonstrated a sign language recognition system limited to a 40 word vocabulary and a controlled environment [9]. We are currently working to extend this research to a sign language recognition system for mobile environments. The aim of the project is to recognize the basic vocabulary and grammatical features found in Contact Sign (a modified subset of American Sign Language) on a mobile computing platform. We will use computer vision, accelerometers, and other mobile sensors for sensing and employ an interface that allows rapid feedback and correction by the user. Our assumptions for the current prototype are: the active participation of the signer in the recognition process and user familiarity with the basic vocabulary and grammar of their target audience. This recognition system could ultimately be extended to act as an input device for phones using Short Messaging Service (SMS) or Telecommunication Device for the Deaf (TDD). The system could also be used as a component in a sign–to–English translation system.

We propose the use of multiple sensor types to allow disambiguation of gestures for recognition. Accelerometers with three degrees of freedom, mounted on the wrists and torso are used to increase our sensing information. The accelerometers will capture information that is difficult for the vision system to obtain, such as rotation (when hand shape looks similar) and vertical movement in the direction of the camera. The camera will provide information not gathered by the accelerometers, such as hand shape and relative position. Both sensors collect information about the movement of the hands through space. It is our goal that by adding multiple sensor types, the accuracy of the system will be improved in noisy or problematic conditions.

Figure 6: View from hat mounted camera with the image defocused to improve tracking. Image superimposed in the upper–left corner shows the participant wearing the hat–mounted camera along with a head–mounted display on her glasses. Note the downward–facing camera mounted beneath the brim of the hat and the head–mounted display over the left eye.

4.3.1 Preparation

Designing Gesture Models: Gestures in the vocabulary were represented by two different HMM topologies. Short gestures {my, me, talk, exit, calibrate} were represented with a five state, left–to–right HMM. A diagram of this model can be seen in Figure 7. Longer gestures {computer, helps} were represented with a ten state, left–to–right HMM with self–transitions and two skip states.

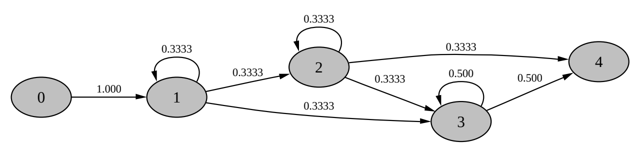

Figure 7: Topology for a five state left to right HMM with self transitions and one skip state

Specifying a Grammar: For this experiment 72 hand–signed sentences were performed. These sentences consisted of permutations of the five–word vocabulary {my, computer, helps, me, talk}. The start and end of each sentence are indicated by special calibration and termination gestures. The grammar used by $\mathbf{GT^2k}$ has the form:

```
word = my | computer | helps | me | talk
sentence = ( calibrate word word word word word exit )
```

Data Collection and Annotation: Our current research system consists of a wearable computer, heads-up display, hat-mounted camera, and accelerometers. The system captures video of the user signing along with accelerometer data from the wrists and body. A sample image from the camera can be seen in Figure 6. The left hand is marked by a cyan band and the right hand is marked by a yellow band. We use multiple sensors to help disambiguate sensing in a noisy

Feature Set	Mean	StdDev
Vision	52.38%	8.97
Accelerometer	65.87%	16.53
Combined	90.48%	11.25

Table 1: System performance on independent vision feature set, independent accelerometer set, and combined feature set

environment. In particular, the vision is susceptible to errors from noise. The results presented here represent a test comparison of validation on vision data, accelerometer data and both data sets combined [3].

The feature sets used for training consisted of accelerometer data and vision data. The accelerometer feature vector consists of (x,y,z) values for each of the three accelerometers. The vision feature vector consists of the following blob characteristics: x,y center coordinates, mass, eccentricity, angle, major axis x,y coordinates, and minor axis x,y coordinates. The camera captures at twelve frames per second and each frame is synchronized with ten to twelve accelerometer packets. The data is combined into a single feature vector. The accelerometer values are an average of the packets that accompany each video frame.

4.3.2 Training, Validation, and System Performance

Leave–one–out validation was used for training and validation of the gesture models. Table 1 shows the average word level accuracy over all of the runs. Experiments were conducted by performing recognition over the training data as well as an independent validation set. These results indicate that the system performance dramatically improves when using the combined accelerometer and vision feature vector over either individual sensor feature vector.

4.3.3 Recognition Application

A real-time recognition system has not yet been implemented. Currently work is being done to expand the vocabulary size and improve recognition rates. However, this pilot project demonstrates that it is possible to recognize sign language on a mobile platform.

4.4 Workshop Activity Recognition

Gesture recognition systems typically rely on deliberate gestures performed by the user as an interface to a system. In some situations it may be beneficial for a system to monitor the typical actions of a user to determine the user's context and react without explicit intervention. For example, the system may monitor an assembly task and alert the assembler when a step is missed. The Workshop Activity Recognition [7] project attempts to recognize common actions a user performs while constructing an object in a workshop. Examples include drilling, sawing, hammering, etc. For this experiment, accelerometers worn on the body, instead of cameras, sense the user's actions.

4.4.1 Preparation

Data Collection: To collect data, a user was asked to perform the same task ten times. The task consisted of assembling a simple object from two pieces of wood and a piece of metal. Each time the object is assembled the same sequence of gestures is performed. This sequence includes examples of the following gestures: hammer, file, sand, saw, screw, drill, clap, use_drawer, and grind. Data is collected using three–axis accelerometers positioned at the wrist and elbow. Sensors were attached to an on–body computer to store the readings. Figure 8 shows sample accelerometer data and an image of the workshop and tools used to collect the data. Each accelerometer returns three values at 100 Hz. Readings from the two accelerometers result in a feature vector of length six.

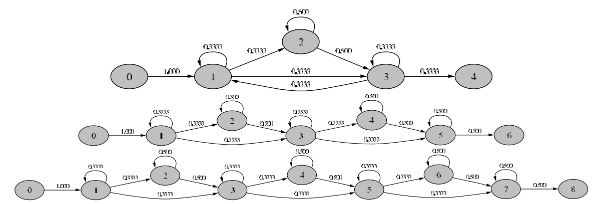

Figure 9: Example workshop HMM topologies

$\mathbf{GT^2k}$ was used to construct models of each gesture based on the accelerometer data. Examples of each gesture were hand-parsed from the ten repetitions of the task.

Designing Gesture Models: We found that it was appropriate to model certain groups of activities using different HMM topologies. For file, sand, saw, and screw, a simple five–state model suffices because they consist of simple repetitive motions. Drill is better represented using a seven–state model. Clapping, use_drawer, and grinding are slightly more complex and required nine–state models. The vise is unique in that it has two separate motions, opening and closing. Thus a 9 state model is used with two appropriate loopbacks to correctly represent the gesture. These models were selected through inspection of the data, an understanding of nature of the activities, and experience with HMMs. Examples can be seen Figure 9.

Specifying a Grammar: Training and recognition in this experiment were performed using isolated gestures. Thus the grammar is simple and was defined as follows:

```
gesture = hammer | file | sand | saw | screw | vise
| drill | clap | use_drawer | grind
```

4.4.2 Training, Validation, and System Performance

Training was performed on isolated examples of each gesture. Leave–one–out validation was performed on the set of training examples. The HMMs were able to correctly identify 93.33% of the gestures over the data collected.

4.4.3 Recognition Application

Currently, a real–time recognition system has not been completed, but future work will involve equipping a user with the hardware and performing the recognition during the performance of the tasks. Further work will also extend the set of gestures beyond workshop activities to everyday gestures such as typing, writing, and walking.

5. FUTURE WORK

The public release date for $\mathbf{GT^2k}$ is planned for Fall 2003. The release will be supported by a web site and mailing lists for community development. The code will be open source and publicly available. In–house projects will contribute feedback for continued development of the toolkit. We also hope to use comments and suggestions from the community to help direct the development of the toolkit.

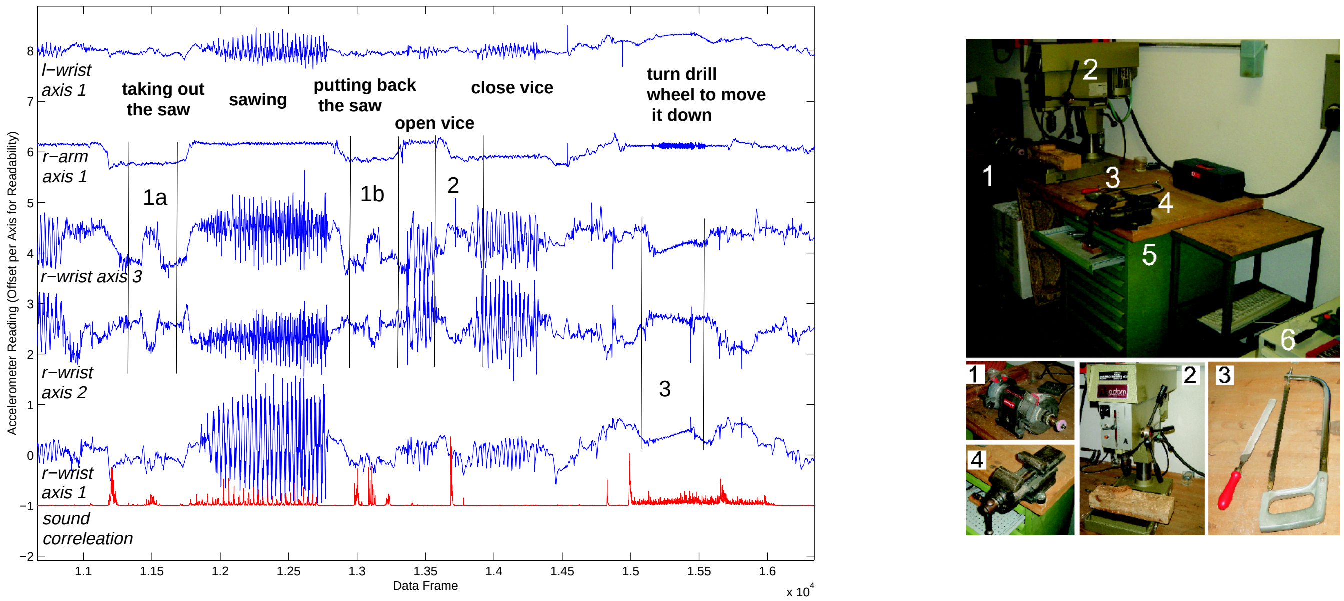

Figure 8: Left: Example accelerometer data from sawing and drilling. All three axes from the right wrist accelerometer and one axis from the right arm accelerometer are shown. The left wrist accelerometer data is not used in this experiment. Right: the wood workshop with 1) grinder, 2) drill, 3)file and saw, 4) vise, and 5) cabinet with drawers.

6. CONCLUSIONS

We have introduced the Georgia Tech Gesture Toolkit, $\mathbf{GT^2k}$. The toolkit is designed to simplify the task of building applications that contain a gesture recognition component. $\mathbf{GT^2k}$ provides a high–level abstraction to the hidden Markov model toolkit (HTK) by supplying the user with a number of tools to create, train, test, and visualize HMMs. We have discussed the use of $\mathbf{GT^2k}$ in system development, and shown its use in four on–going research domains.

7. ACKNOWLEDGMENTS

Funding, in part, from the Graphics, Visualization and Usability Center Seed Grant program, by NSF career grant # 0093291 and by the Rehabilitation Engineering Research Center on Mobile Wireless Technologies for Persons with Disabilities (Wireless RERC).

We would like to thank our colleagues Holger Junker, Paul Lukowicz, and Gerhard Tröster at the Swiss Federal Institute of Technology(ETHZ) for the use of their accelerometer package. We would also like to thank our sponsors at Visteon and DaimlerChrysler for their support.

The opinions contained in this publication are those of the authors and do not necessarily reflect those of the U.S. Department of Education, the National Science Foundation and other funding institutions.

8. REFERENCES

[1] HTK Hidden Markov Model Toolkit home page. http://htk.eng.cam.ac.uk/.

[2] M. Betke, W. Mullally, and J. Magee. Active detection of eye scleras in real time. In *Proceedings of the IEEE CVPR Workshop on Human Modeling, Analysis and Synthesis*, June 2000.

[3] H. Brashear, T. Starner, P. Lukowicz, and H. Junker. Using multiple sensors for mobile sign language recognition. In *Proceedings of IEEE International Symposium on Wearable Computing*, page In Press, October 2003.

[4] P. de la Hamette, P. Lukowicz, G. Tröster, and T. Svoboda. Fingermouse: A wearable hand tracking system. In *Fourth International Conference on Ubiquitous Computing*, pages 15–16, September 2002.

[5] R. Liang and M. Ouhyoung. A real-time continuous gesture interface for Taiwanese Sign Language. In *Submitted to UIST*, 1997.

[6] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI81*, pages 674–679, 1981.

[7] P. Lukowicz, J. Ward, H. Junker, M. Stäger, G. Tröster, A. Atrash, and T. Starner. Recognizing workshop activity using body worn microphones and accelerometers. Technical Report GIT-GVU-03-32, Georgia Institute of Technology, 2003.

[8] L. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, Feb 1989.

[9] T. Starner, J. Weaver, and A. Pentland. Real-time American Sign Language recognition using desk and wearable computer-based video. *IEEE Trans. Patt. Analy. and Mach. Intell.*, 20(12), December 1998.

[10] C. Vogler and D. Metaxas. Adapting Hidden Markov Models for ASL recognition by using three-dimensional computer vision methods. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pages 156–161, Orlando, FL, October 1997.

[11] S. Young. The HTK Hidden Markov Model Toolkit: Design and Philosophy, 1993.