Improving fluency in sign language to text systems

Sam Black
524689

Supervisor
Prof Martin Russell

Final Report

Table of Contents

1 Introduction

Communication via sign language is more universal than the spoken word; we all understand what is meant by someone pointing, covering our ears over with our hands, or motioning somewhere.

Automatic sign language recognition (ASLR) systems lag some way behind other recognition systems for the simple reason that gesture permutations are nearly infinite; ask someone to move their hand from above their head to by their waist and you'll get variations every time. Capturing the motion of the person's arm in an non-intrusive way also limits the usefulness of a system, with the least intrusive systems being the most complex to implement.

To account for the first problem, a statistical model of the various motions in the form of a Hidden Markov Model (HMM) is used; this model maps the most likely transition of the person's arms signifying a word, and because it works using probabilities, it can be designed to allow for minor variations between each arm movement, user or application. The second problem is more difficult to overcome, and will be the subject of many other research projects and advances in technology.

Whilst research in ASLR has been conducted, previous systems to convert sign language to text have concentrated on translating individual letters rather than using whole words; whilst this limits the corpus needed, it is uncomfortable and inconvenient to spell each word rather than just sign it (imagine phonetically spelling each word). Most systems currently cannot translate sign language to text in real time fluently, requiring the user to sign each letter or word discretely, reducing the usefulness of the system in a real world setting.

Thus, the aim of the system is to use HMM to implement a fluent British sign language (BSL) to text translator for a limited corpus, such as an information help point for the University campus (for example, “Where is the Guild?”).

2 Research

Automatic sign language recognition (ASLR) is a combination of gesture recognition and facial recognition (Edwards, 1997). Various methods exist to capture the data, most systems utilise an image capture system that extracts the vector data. The unobtrusive nature of this method is the major factor to its wider adoption and basis of research. Image capture is less accurate than using a motion tracking system, such as using a data glove or optical motion tracking system, as there is a lower amount of noise and segmentation of the data (Dreuw, Rybach, Deselaers, *et al*, 2007).

Accuracy in the system is approaching that of automatic speech recognition (ASR) systems of 85% - 90% (Holt, Hendriks, and Andringa, pp. 8), but lower word error rates (WER) can be achieved by using multiple layers of HMMs. Layered HMM are best for systems which require low response times with a large corpus (Zhang, Yao, Jiang, *et al*, 2005).

Whilst other systems are relatively successful with identifying finger spelt words (Travieso, Alonso, and Ferrer, 2003), discrete isolated words signed (Grobel and Assan, 1997) with no attempt at grammatically correct sentence structure (Akmeliawati, Ooi and Kuang, 2007), few are successful in fluent sentence recognition with a grammar. Computer processed visual data is still susceptible to noise and interference, such as the background being of similar colour to the users hands, adverse lighting conditions or other people in motion providing spurious inputs (Je, Kim and Kim, 2007). Computer vision based systems offer the most comprehensive coverage of a sign language user's movements, but are computationally expensive to calculate the vectors off the user's body and limbs (Holt, Hendriks, and Andringa, pp 11).

3 Projected Outcomes

To construct a fluent, real-time ASLR system based on HMM, using a motion tracker to collect the sign data for PCA. As an extension, enhance the system to work with a camera or webcam to capture data.

4 Requirements

The practical requirements to achieve the outcomes are listed below;

1. Create the corpus to be signed
2. Create grammar, models and pronunciation dictionary for HTK
3. Record the signing using a motion tracker
4. Compile the vector data from the motion tracker
5. Conduct principle component analysis (PCA) on the vectors
6. Run training data through HTK
7. Improve the grammar, models and pronunciation dictionary
8. Re-run the training data to test improvements
9. Refactor code to run in real time

5 System Construction

The system is to recognise BSL sentences related to interacting with the information points around the campus.

The system has to take in 3D positional data obtained from a motion tracker and either use it to create the HMMs or to translate the gesture data into text.

The system would have to be broken down into 4 parts;

- Phrases to be signed
- Motion data capture
- Hidden Markov Models
- Software implementation

5.1 Phrases to be signed

Creating the phrase list to be signed required that each word used have about 8 occurrences in across all the phrases for accurate Markov models to be created.

Two BSL signers would sign 40 phrases each (although in practice one signer completed only 30 phrases), each signer repeating the phrase twice with a “silence” in between each phrase. The “silence” would be the signer putting their hands by their sides.

The phrase list can be found in Appendix 10.2, with the word list in Appendix 10.3.

5.2 Motion data capture

To create an accurate system, the motion data had to be as comprehensive as possible.

The tracking system would record 30 seconds of data at 200Hz, with the signer signing each phrase twice, for example 'pause, “What time is it”, pause, “What time is it”, pause'. A sample of the timings for a phrase can be found in Appendix 10.4.

The signer's movements would be tracked using an infra-red based camera system, where reflective markers were placed on the signer's fingers, hands, arms and waist. The system then tracked the motion of the markers and recorded the 3D positional data of each marker.

The motion tracking system, as with any other optical based system, suffers from occlusion, where some markers would be invisible to all the cameras all of the time, creating duplicate markers. In some instances the markers were out of sight so frequently that they were tracked for about 10% of the time, adding another 250 markers to the tracking data.

Some of these limitations can be overcome by labelling the markers and linking them via “bones” to create a wireframe model of the signer and applying this model to the tracking data to help remove the duplicates of the markers. This model, alongside a normal photograph of the signer, can be seen in Figures 1 and 2.

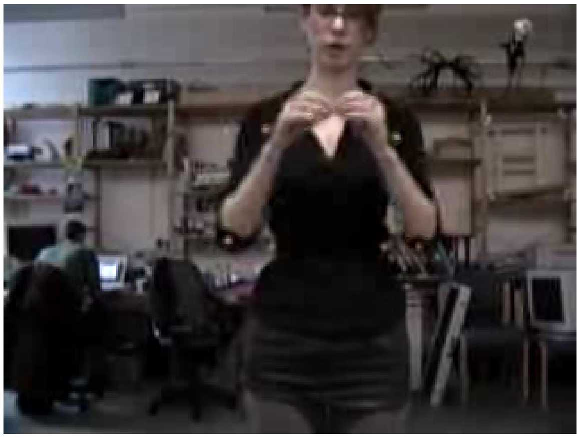


Figure 1: Signer with reflective markers

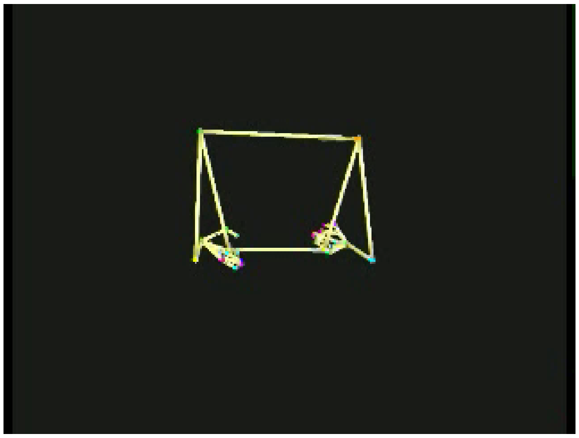


Figure 2: Wireframe model

The wireframe model does not remove all problems however; the tracking system can extrapolate the position of the markers, using the wireframe model as a guide.

Ideally, since the length of the arms, finger tips to finger joints and shoulder to waist measurements do not change, the system should correctly extrapolate any missing positional data. This was not the case however, as can be seen in Figures 3 and 5.

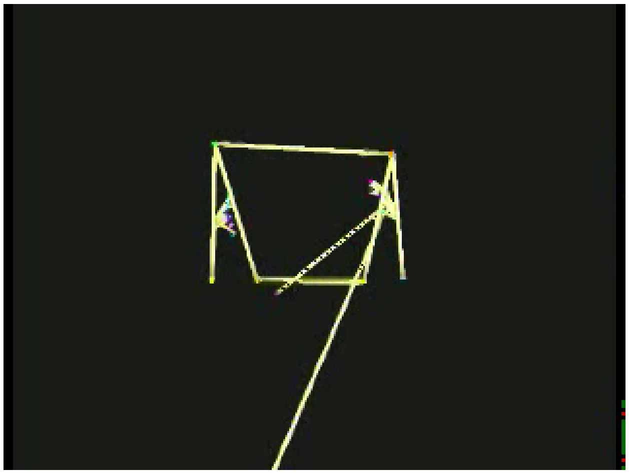


Figure 3: Incorrect extrapolation

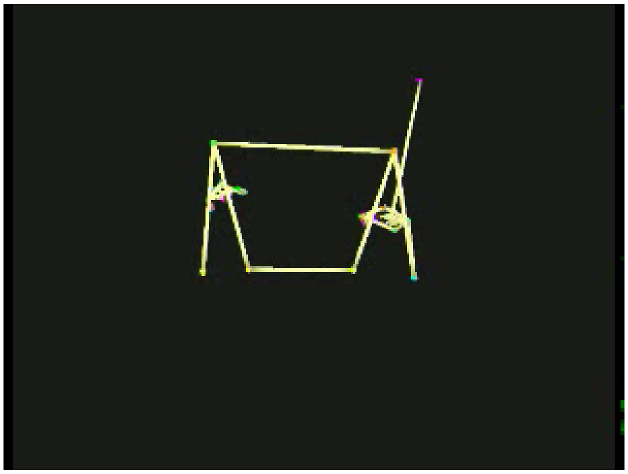


Figure 4: Incorrect extrapolation

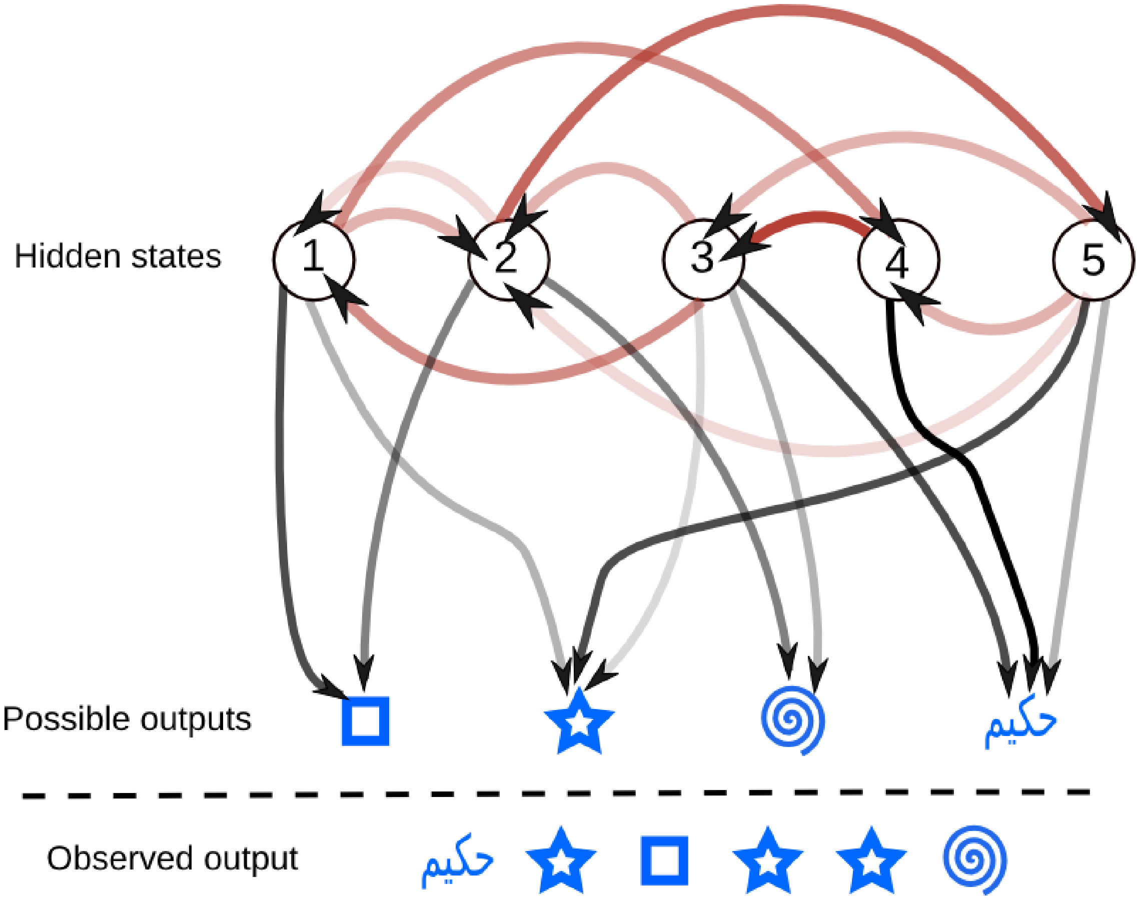
The only way to correct this would be to manually label every marker tracked by the system which would be prohibitively time consuming, or to allow some data to be lost and lower the accuracy of the ASLR system.

5.3 Hidden Markov Models

Hidden Markov Models (HMMs) can be used to determine a sequence of hidden states based on the observed output using the probabilities of state transitions and state emissions.

Initially, the starting hidden state, the number of hidden states, the state transition probabilities and the emission probabilities are all unknown; only the observed output sequence is known. To be able to construct a HMM, the observed output has to be labelled; for a signed sequence of “what time is it?”, the recorded signing could be labelled as [silence, “what”, “time”, “is it”, silence], each label having the start and end times of the word signed. Using this information, the HMM can be constructed using the Baum-Welch algorithm, which takes the positional data labelled (eg for “what”) and estimates the number of hidden states associated for “what”, and then estimates the associated probabilities. If another occurrence of “what” is found in a different sentence, the Baum-Welch algorithm modifies the initial estimate to fit the new data; eventually the algorithm iterates over all instances of “what” to construct a model based on the provided training data.

Figure 5 illustrates how the most likely HMM would appear for an arbitrary set of data after the Baum-Welch algorithm has constructed it using the training data.



Original drawing by Hakeem Gadi
Creative Commons Attribution ShareAlike 3.0 Licence

Figure 5: HMM structure

In this example, the outputs (a square, star, spiral and Arabic letter) are arbitrary labels for illustrative purposes only.

The red lines denote transitions from one state to another, with same state transition lines (such as transitioning from state 5 to state 5) omitted for clarity. The black lines denote which output each hidden state can emit.

The opacity of the lines signifies the relative probability of each action, the less opaque lines being low probability and the more opaque being high probabilities.

After the HMM has been constructed, an observed output sequence can be decoded using the Viterbi algorithm.

The Viterbi algorithm takes the state transitions and emissions probabilities and traverses each possible path from state to state that could create the observed outputs, ranking each path by probability; the highest ranking path would be returned as the most likely state sequence to produce the observed output sequence.

In the example above, the most likely sequence could be 5 (emits Arabic letter), 5 (emits star), 2 (emits square), 5 (emits star), 5 (emits star) then 3 (emits spiral), where the hidden sequence would be 5,5,2,5,5,3.

For the ASLR system, the hidden states would correspond to the words signed and the observed outputs are the recorded positional data values of the gestures for each word.

5.4 Software implementation

HTK is a collection of programs designed to take annotated audio files, construct HMMs and then recognise speech.

HTK can be adapted to recognise any sequence of data, and could be used to recognise gestures, handwriting, music scores or for bioinformatics.

The system application is written in python, which wraps around the HTK applications needed to create the HMMs, to test and evaluate them. The application is configured using text files to list the data files, annotations, output files and HTK arguments to use.

The 3D motion tracking data is separated into training, testing and evaluation sets, roughly 50%, 20% and 30% respectively. This data is converted from the raw data (a sample of which is in Appendix 10.5) into the HTK file format; the HTK file format consists of a header describing the format of the data, the sample rate, the number of samples and the data itself. This is written directly to file as binary output of each component.

The raw data to HTK file format did not work initially, as the system was reading in the tracker data files that had had the model applied incorrectly, resulting in the 200+ markers as outlined above. This produced a 600 column (200 X, Y and Z co-ordinate sets), 6000 row (for 30 seconds of recorded data at 200 Hz) file. The HTK file format application effectively tried to create and write out to file a 600 by 6000 floating point array, causing the application to crash. Correctly applying the wireframe model to the data set corrected this issue.

6 Achievements

The achievements based on the project requirements are listed below, with a system overview provided in Figure 6;

1. Create the corpus to be signed; **Completed**
2. Create grammar, models and pronunciation dictionary for HTK; **Completed**
3. Record the signing using a motion tracker; **Completed**
4. Compile the vector data from the motion tracker; **Incomplete**
5. Conduct principle component analysis (PCA) on the vectors; **PCA software written**
6. Run training data through HTK; **HTK Python wrappers written**
7. Improve the grammar, models and pronunciation dictionary; **Incomplete**
8. Re-run the training data to test improvements; **Incomplete**
9. Refactor code to run in real time; **Incomplete**

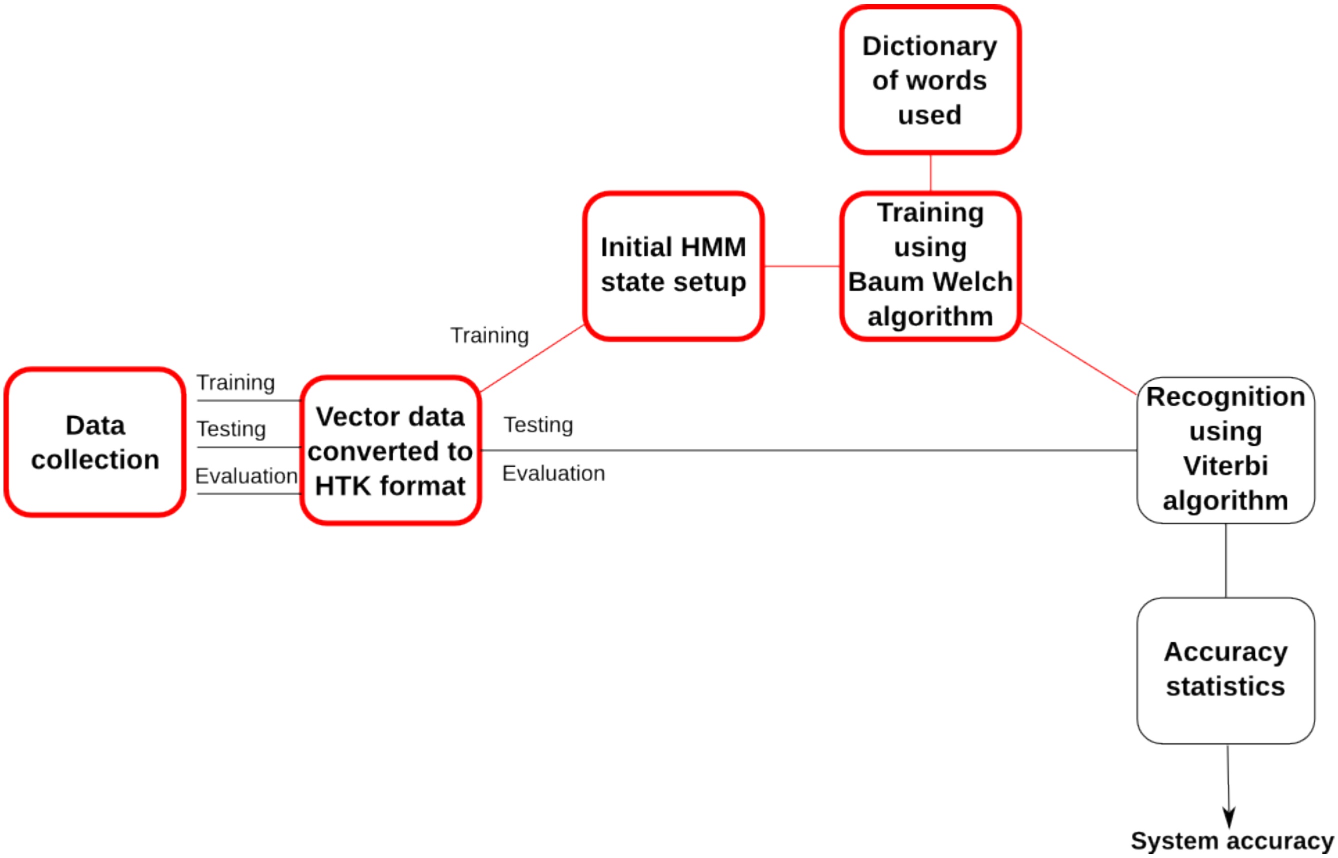


Figure 6: System components block diagram. Red boxes denote completed sections

7 Conclusions

As shown in Figure 6, the system is in a ready state to begin processing 3D tracking data, save for any bugs in the software that would be exposed whilst using the whole system.

The 3D tracking data however, as outlined in section 5.2, took much longer than expected to apply the wireframe model in a consistent and useful manner, resulting in the project being incomplete at the current time.

The project could be implemented given another few weeks of reapplying the wireframe model, however this would result in poor tracking data (as shown in Figures 3 and 4). Removing these anomalies completely in the data files by accurately labelling every marker recorded by the tracking system could take months to complete.

Whilst some data is correctly labelled, there is an insufficient amount of it to be able to construct an HMM from it; the system would be creating a statistical model based on one or two data sets only, so that the recognition and evaluation modules would not correctly identify the gestures tested.

For example, the BSL gesture for “time” is to point at your left wrist. With a full training set, the system would be able to recognise the “time” gesture regardless of the speed of the gesture, variations in where the signer points or where their left hand is with respect to the rest of their body and so on; with a limited or singular training data set, the system would only recognise that gesture if it matched the specific 3D positional data exactly. This would not occur because the training data is not used in the testing, as outlined in section 5.4.

Overall therefore, the project was left incomplete due to underestimating the time required to create accurate and reliable 3D tracking data. This could be remedied by recapturing the gesture data with an emphasis on ensuring that the cameras of the tracking system are positioned better to limit the amount of occlusion and data loss occurring, and to allocate more time to applying the wireframe model to all the data collected.

8 Acknowledgements

Many thanks to the signers, Grace Sheppard and Jenny Warr, and also to Professor Martin Russell for support and guidance throughout the project.

9 References

Akmeliawati, R. Ooi, M. P. L. and Kuang, Y. C. (2007). **Real-Time Malaysian Sign Language Translation using Colour Segmentation and Neural Network.** Proceedings of Instrumentation and Measurement Technology Conference, Warsaw, Poland, 1-3 May 2007.

Dreuw, P. Rybach, D. Deselaers, *D. et al.* (2007). **Speech Recognition Techniques for a Sign Language Recognition System.** In Interspeech, pages 2513-2516, Antwerp, Belgium, August 2007.

Edwards, A. D. N. (1997). If the glove fits: Progress in sign language recognition. **Ability**, 22: pp. 12-13

Grobel, K. and Assan, M. (1997) **Isolated Sign Language Recognition using Hidden Markov Models.** Systems, Man, and Cybernetics, 'Computational Cybernetics and Simulation'. IEEE International Conference 12-15 Oct. 1997. Volume 1, pp. 162-167

t. Holt, G. Hendriks, P. and Andringa T. (unknown date). **Why don't you see what I mean?** [online]. http://ict.ewi.tudelft.nl/pub/gineke/ASR05.pdf [Accessed 07 October 2008]

Je, H. Kim, J. and Kim, D. (2007). **Vision-Based Hand Gesture Recognition for Understanding Musical Time Pattern and Tempo.** In Industrial Electronics Society, 33rd Annual Conference of the IEEE, 5-8 Nov. 2007, pp. 2371-2376

Travieso, C.M. Alonso, J.B. and Ferrer, M.A. (2003). **Sign language to text by SVM.** Proceedings of the Signal Processing and Its Applications, Seventh International Symposium 1-4 July 2003, Volume 2, pp. 435-438

Zhang, C. Yao, H. Jiang, F. *et al* (2005). **Multilayer Method Based On Multi-Resolution Feature Extracting and MVC Dimension Reducing Method for Sign Language Recognition.** Proceedings of the Fourth International Conference on Machine Learning and Cybernetics, Guangzhou, 18-21 August 2005

10 Appendices

10.1 Source code

The HTK source code, research, reports, language models, grammars, dictionaries and scripts are included on the attached CD, and is also accessible using source code management software package GIT (http://git-scm.com/) by;

```
     git clone http://repos.lapwing.homelinux.org/personal-uni.git
```

and can be update by

```
     git pull
```

in the directory **personal-uni/**.

Source code and documents for this project is in the personal-uni/ee4p/ directory.

10.2 Script for signing

Where is the nearest telephone?
Where is the Gisbert Kapp building?
Where can I get a taxi?
When is the next bus to the Vale?
Which is the quickest way to University Centre?
The quickest route to the Baber Institute.
What is the quickest route from Gisbert Kapp to the Clock Tower?
Where is the bus station?
What time is the Guild open?
The Gisbert Kapp building is 50
Where is the Baber Institute?
What is the quickest route to the train station?
Is there a way to the telephone?
What is north of the University?
Where is the nearest bus station?
Where is the Bristol Road?
Which road is near the Guild?
How far away is the Gisbert Kapp building?
How near is the Baber Institute?
What is the quickest route from the telephone to the cash machine?
Where is the nearest cash machine?
How far to the centre from the University?
Where is the train station?
When is the next bus?

Where is the nearest bus station to the Baber Institute?
What road is near the Gisbert Kapp building?
Which building is nearest to the Baber Institute?
The Baber Institute building nearest the machine
What is the route from the road to the car park?
What is near the Clock Tower?
Is the telephone near the Baber Institute?
Where is University Centre?
Near University Centre.
What machine is in the Gisbert Kapp?
Where is the bus station?
Where is the cash machine nearest the road?
Where is the nearest taxi?
When is the bus?
How far from the University to the Vale?
What is the building nearest the road?

The phrases in italics were only completed by one signer and were to be used as the evaluation data set.

10.3 Word list

the
is
to
where
what
nearest
road
building
university
bus
gisbert
centre
near
kapp
train
station
vale
next
tower
quickest
in
how
guild

from
when
institute
clock
which
park
baber
machine
far
get
car
there
cash
open
I
can
way
a
north
telephone
route
away
taxi
time
of
bristol

10.4 Annotated data sample

Timings are in 100ns separations, the default for HTK.

`sil` represents silence, where the signer put their hands by their side.

```
"*/grace_script_03.lab"
000000 1345000 sil
1345000 2989000 taxi
2989000 4011000 where
4011000 5132000 sil
5132000 6875000 taxi
6875000 8096000 where
8096000 34326349 sil
.
```

10.5 3D motion tracking output data sample

NO_OF_FRAMES	6000								
NO_OF_CAMERAS	12								
NO_OF_MARKERS	28								
FREQUENCY	200								
NO_OF_ANALOG	0								
ANALOG_FREQUENCY	0								
DESCRIPTION	--								
TIME_STAMP	12/12/08	13:25:29	3754.358						
DATA_INCLUDED	3D								
MARKER_NAMES	Left Shoulder			Left waist			Left elbow		
	-286.868	-244.932	560.569	-216.844	-161.473	301.485	-273.988	-245.494	252.796
	-285.436	-241.649	558.029	-217.517	-162.049	300.063	-274.175	-244.39	251.86
	-285.269	-241.618	559.39	-218.07	-162.87	299.168	-274.133	-245.158	252.781
	-285.851	-241.222	559.056	-217.042	-161.463	301.045	-274.171	-245.131	252.781

Not all data from the file is included, a sample of the collected data can be found on the CD.