Real-Time Hand Tracking and Gesture Recognition Using Smart Snakes

Tony Heap*and Ferdinando Samaria

Olivetti Research Limited
Old Addenbrookes Site, 24a Trumpington St.
Cambridge CB2 1QA, United Kingdom, tel.+44 1223 343000

June 20, 1995

Abstract

This paper gives architecture and implementation details of a hand tracking and gesture recognition system which has been developed at Olivetti Research Limited as an application for the Medusa distributed multimedia environment. The system uses live 15-bit colour video from a networked camera, runs in real time (25 frames/sec on a DEC Alpha), and copes well with background clutter. Tracking is achieved using the 2D deformable Active Shape Models (smart snakes) of Cootes and Taylor, and a genetic algorithm is used to perform an initial global image search. The Point Distribution Model used for both snakes and genetic algorithms provides a generic and flexible model that can be used to track any 2D deformable object.

1 Introduction

Our hands play a very important role in communication. Not only do we specify position and movement with them, but we perform meaningful gestures such as thumbs up for 'yes' or a palm for 'stop'. Current technology does not capture the full potential for hand-based interaction. The mouse restricts movement to two dimensions, with button presses being the only available gestures. The dataglove provides full 3D control, but the need to continuously put on and remove the glove, the intrusive cabling and the

*Now working in the School of Computer Studies, University of Leeds (ajh@scs.leeds.ac.uk)

restrictively expensive hardware make it suitable only for specialized tasks. A possible alternative is to use a vision-based system.

The ability to track a human hand and recognise any gestures made, using only visual input, is taken entirely for granted by humans, but is an inherently difficult task to perform for a computer. As well as being able to move with 6 degrees of freedom in 3 dimensional space, a total of 15 finger joints allow for 21 degrees of freedom in terms of deformation of shape. Add to this the variation in shape from person to person, and the complexity of the task becomes apparent.

Several attempts have been made to produce systems which perform such a task. Dorner [1] designed a system for interpreting American Sign Language. This worked in 3D, but the user had to wear a glove with coloured markings in order to simplify feature extraction, and real-time execution was not possible. Rehg and Kanade [2] designed a system called DigitEyes, which modelled the hand as jointed cylinders, and used stereo vision to track a hand in 3 dimensions, using only natural image features such as edges. Performance was good, but a controlled background was required, making its practical use limited. Blake *et al* [3] describe a robust real-time system based on contour tracking; this works well, but treats the hand as a rigid body, capable only of affine transformation, so changes in *articulation* are not recognised. Kervrann and Heitz [4] combine Blake's approach with 2D deformable models to give extra functionality, but they have not yet achieved real-time performance.

Whilst all these systems go some way towards achieving the stated goal, none of them manages to combine *all* the features necessary for Human-Computer Interaction purposes.

This paper describes the architecture and implementation of a system which extends some way towards this goal. Its chief limitation is that it is only capable of tracking an open hand; the model used is only 2D. However, it does work in real-time (over 25 frames/second on a standard DEC Alpha UNIX workstation), and it copes well with cluttered backgrounds and variation in lighting conditions.

The system is based around the 2D deformable Active Shape Models (or Smart Snakes) of Cootes and Taylor [5]; a deformable outline model tracks a hand in an image by being 'attracted' to edges in the image. The hand model is also used in combination with a genetic algorithm to perform an initial image search in order to locate the hand, as suggested by Hill [6].

The system runs under Olivetti's Medusa distributed multimedia platform [7, 8], providing a flexible and portable environment in which any one of a number of ATM networked cameras can be used to provide 15-bit colour video sequences.

The remainder of the paper is organised as follows: section 2 describes the use of Smart Snakes for tracking; section 3 describes the use of genetic algorithms for the initial image search; section 4 discusses scene illumination issues; section 5 describes the implementation of two application programs, and finally, section 5 draws some conclusions.

2 Feature Tracking Using Smart Snakes

Active Shape Models (ASMs or Smart Snakes) were originally designed as a method for exactly locating a feature within a still image, given a good initial guess. The concept used is straightforward. A contour which is roughly the shape of the feature to be located is placed on the image, close to the feature. The contour is attracted to nearby edges (changes in intensity) in the image and can be made to move towards these edges, deforming (within constraints) to exactly fit the feature. The process is iterative, with the contour moving in very small steps.

This technique can be extended to track features across frames of video - the position of a feature in one frame is used as a starting approximation for the next frame.

The success of such a technique relies on the construction of a suitable model of the object to be tracked. Generating a rigid outline for a hand in any particular pose is not a problem, but capturing the way in which the shape can vary, whilst disallowing invalid deformations (this is essential for a robust system) is more difficult. Such a model would also appear to be inherently application-specific, requiring a great deal of work to provide a new model for, say, head tracking.

The Point Distribution Model provides a way of solving both of these problems.

2.1 Overview of the Point Distribution Model

The Point Distribution Model (PDM) encapsulates variation information and can be created using a training process which requires for input a set of examples of the object to be modelled, in various poses. The training examples must be annotated with certain landmark points, being points of interest on the object[1]. By consistently labelling these points over the whole training set, statistical analysis can be performed in order to discover the average model shape, and also the principal modes of variation. Figure 1 gives an outline of the processes involved.

The mean shape is described in terms of the average (x, y) coordinates of the landmark points, after aligning the training examples. These can be combined into a single column vector $\overline{\mathbf{x}}$:

$$\overline{\mathbf{x}} = (x_1, y_1, \ldots, x_n, y_n) \tag{1}$$

where n is the number of points in the model.

The modes of variation are described by sets of x and y displacements for the landmark points, which can be combined into column vectors $\mathbf{v_i}$:

[1]Note that landmark points were initially entered by hand, using the X-Windows™ application, **xv**, to obtain (x, y) coordinates. A method for extracting the points automatically from images has also been developed.

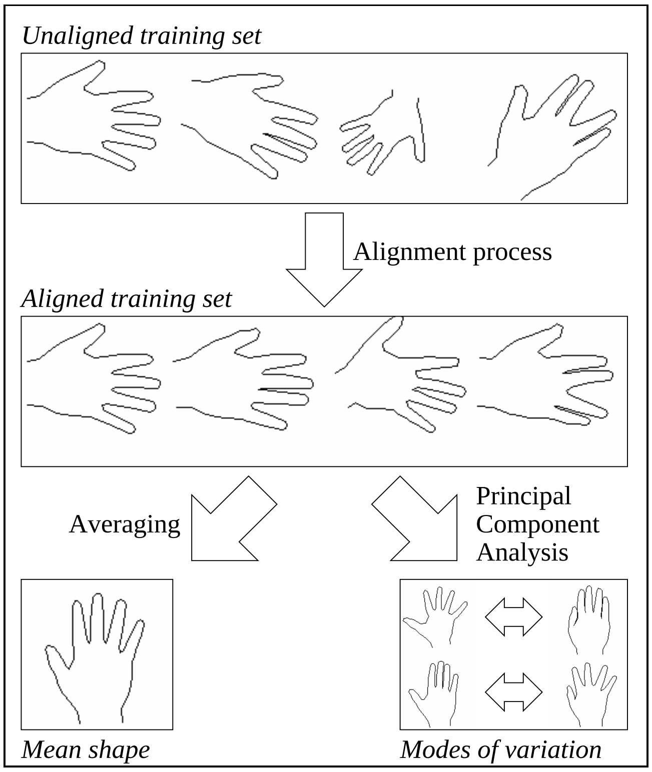

Figure 1: Training a Point Distribution Model

$$\mathbf{v_i} = (dx_{i1}, dy_{i1}, \ldots, dx_{ik}, dy_{ik}, \ldots, dx_{in}, dy_{in}) \tag{2}$$

where $1 \leq i \leq n$, so the i^{th} variation vector affects point k in the model by moving it along a vector parallel to (dx_{ik}, dy_{ik}).

A general model shape $\mathbf{x}$ is generated by adding linear combinations of the most significant variation vectors to the mean shape:

$$\mathbf{x} = \overline{\mathbf{x}} + \mathbf{V}\mathbf{w} \tag{3}$$

where $\mathbf{V} = (\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_t})$ is the matrix of the t most significant variation vectors, and $\mathbf{w} = (w_1, w_2, \ldots, w_t)$ is a vector of weights for the individual variation vectors. Different instances of the model are generated by varying the weights w_i. Invalid deformations are avoided by keeping the w_i within suitable limits, and by ignoring the variation vectors $\mathbf{v_{t+1}}$ onwards, which tend to describe noise in the training data.

To illustrate this concept, Figures 2 and 3 demonstrate the effect on the model shape of varying w_1 and w_2, whilst keeping all other w_i at zero.

The model can be projected onto an image by specifying its location, in terms of scale s, rotation θ, x-translation t_x and y translation t_y, and its pose in terms of the variation vector weights w_i

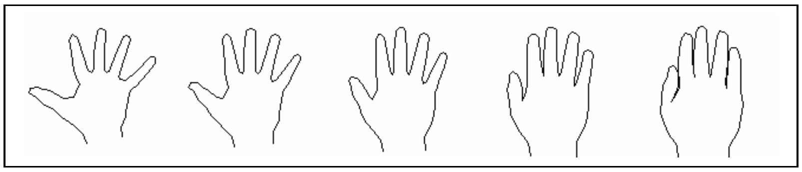

Figure 2: The first mode of variation (w_1 = -100, -50, 0, +50, +100)

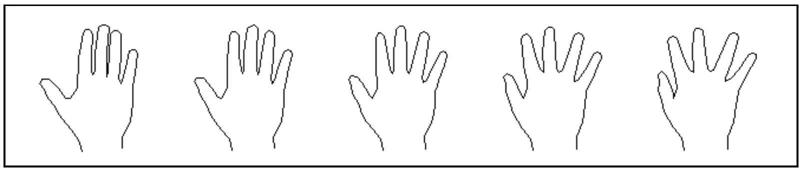

Figure 3: The second mode of variation (w_2 = -100, -50, 0, +50, +100)

2.2 Feature Tracking

As mentioned above, tracking is achieved via an iterative process in which the ASM is attracted to edges in the image. Following edges is straightforward to implement; providing the model is nearby the new edge position, the direction of movement can be found by examining lines of pixels normal to the model boundary in either direction to look for the strongest change in pixel intensity (see Figure 4).

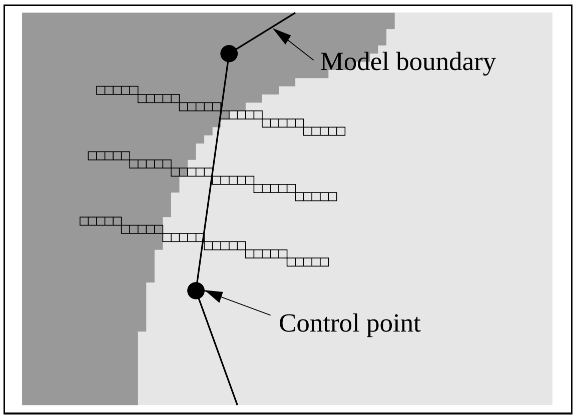

Figure 4: Inspecting pixels along the boundary normal

However, the boundary movement must be constrained to occur only within the allowable model deformation. Given that the model's current projection onto the image is specified by s, θ, t_x, t_y and $w_1 \ldots w_t$, the following process is used to provide a new position:

1. For each model point i, analyse pixels normal to the model boundary, looking for the strongest edge. Suggest values for dx_i and dy_i, the movement of the point in the x and y directions, based on this search.

2. Calculate $\overline{dx}$ and $\overline{dy}$, the average translation over all model points. Use these values to translate the whole model ie. update t_x and t_y.

3. Subtract the effects of this translation from all the dx_i and dy_i. This will leave residual values, which must be somehow satisfied.

4. Calculate a suggested overall average scale, $\overline{ds}$, and rotation $\overline{d\theta}$, and use them to update s and θ.

5. Again, subtract the effects of this overall scale and rotation from all the dx_i and dy_i.

6. Any remaining residual required movements are accommodated by deforming the model within constraints; the w_i parameters are adjusted accordingly.

The algorithm is used iteratively in order to converge on a stable solution. It should be noted that the chief advantage of this process is that it is fast, and that complexity is linear in the number of model points.

2.3 Gesture Recognition

The use of the PDM simplifies the task of gesture recognition, because the hand's pose is encapsulated by very few parameters. t_x and t_y give the hand's x and y coordinates, and its rotation θ is also available. Monitoring these parameters over time gives details of how the hand is moving.

Also, the most important modes of *deformation* of the hand are represented by only a few values $w_1 \ldots w_t$; again, these can be monitored over time in order to recognise articulation. It is useful to know how the values of the w_i correspond to particular physical deformations (eg. the moving of the thumb, for a hand model). This can be achieved in simple cases by observing the values manually whilst the tracker is in operation or, in cases where the dependencies are not so obvious, there is the possibility of training a classifying neural net in order to recognise or classify gestures.

3 Image Search Using Genetic Algorithms

One of the main features of an ASM is that it can only correctly locate an image feature if it is given a good starting approximation. It is thus necessary to perform global image search in order to approximately locate the feature before ASM tracking can begin.

Since we have already developed a system for modelling image features, it seems sensible to make use of this capability in our global image search. We develop a *fitness function*; given a hypothesis that the image feature is present in a certain location and pose, in terms of scale, rotation, translation and deformation, how much evidence is there in the image to support this hypothesis? The evidence can be collected (eg. using edge detection) and evaluated to give a numerical *fitness* value for a particular set of pose parameters. The problem is then reduced to one of finding parameter values which maximise the fitness function.

As can be imagined, this function is not particularly well-behaved (it is not differentiable, for example), so traditional analytical methods are unlikely to provide good results. The method we use is based on Genetic Algorithms.

3.1 Overview of Genetic Algorithms

Genetic algorithms (GAs) provide a way of finding maxima in noisy functions, where traditional methods can fail. They are based loosely on the Darwinian theory of evolution - survival of the fittest. Given a function f, for which a maximum is sought, a population of numerical *genes* (potential maxima) is created randomly. For each gene x_i, the *fitness* value $f(x_i)$ is calculated. Genes are then *regenerated* probablistically - in the next generation, genes for which the fitness value is large are strong and will survive, and genes for which it is small are weak and will die. And as in evolution, the processes of *crossover* (mating) and *mutation* are used in an attempt to generate a broad spectrum of genes.

Gene crossover consists of taking two parent genes (thought of as binary strings), performing a splice operation somewhere (randomly) and swapping the two tails over, as shown in Figure 5.

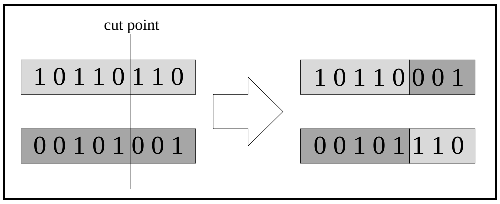

Figure 5: Gene crossover

Mutation consists of occasionally introducing random bit changes (this should occur roughly one time in a thousand).

The process is run for a few hundred generations (iterations) and hopefully by then, the fittest genes will dominate, giving a few good suggestions for x, the maximising argument for f.

Genetic algorithms can succeed where traditional methods may fail, however they are by their nature *probablistic* as opposed to *deterministic*; many different solutions can be obtained for a given problem, and this it is never possible to guarantee 100% success.

3.2 Applying GAs to Image Search

Genetic algorithms can be applied in a straightforward manner to image search. The technique used is based on a system used by Hill [6]. The genes in the population are guesses as to the object location within the image, encoded in integers. The fitness function $f(x)$ gives each guess a score indicating how much evidence there is in the image that the guess is a good one, using an edge detection algorithm. With the gene set and the fitness function defined, the GA search can be executed for a set number of iterations and at the end, the best guesses produced will all provide a close estimate as to the location of one or more instances of the object in the image.

The Point Distribution Model is used as the basis for providing guesses at object location. The PDM parameters of scale, rotation, translation and deformation are coded into a single 32-bit integer, which can be used directly as a bit-string gene in the GA search.

The *fitness* of a particular guess is found by examining short rows of pixels normal to the model boundary. At each model control point, the magnitude of the strongest edge is found, and weighted according to its distance from the boundary (a larger weight for edges close to the boundary). The values for each control point are summed to give an overall fitness.

4 Lighting Issues

One of the main *initial* problems encountered concerned lighting conditions. Human eyes easily adjust to different intensities of light; we can also compensate for scenes which contain a large amount of contrast, use colour information to help recognise objects in a scene, and auto-focus with ease.

Modern video cameras have automatic focussing and brightness control. However, they do not cope well with high contrast images. For example, if one sits directly in front of a window, the light from outside causes the auto-brightness control to decrease light intake, so objects in the foreground become very dark and hence difficult to see. Also, in the common case where the camera is positioned atop a workstation monitor, a hand held up to the camera has very little direct light falling on it, unless there is a window directly behind the monitor, and so is difficult to pick out. Also, background objects can be very distractive, especially white monitor screen casings, which have very sharp edges.

Two measures were implemented in order to improve the performance of the tracker; one physical and one algorithmic.

The physical measure was to install up-lighters either side of the monitor, with white reflective cards angled in such a way as to cast diffuse light onto the face of the user (this is less extreme than having a spotlight shining in one's eyes). This considerably improved overall picture quality. Figure 6 gives *before* and *after* examples (the greyscale image shown here does not do justice to the full colour version, but note that the hand appears lighter in the right-hand image).

Figure 6: Images without (left) and with (right) uplighters

The algorithmic measure was to adjust the relative levels of red, green and blue present in the image. A hand (when properly lit) has a large red component, so red was emphasized. The best balance was actually found to be with *negative* weights for green and blue components. This caused white objects in the background, such as monitor screen casings, to disappear, whilst leaving flesh tones intact. Figure 7 shows the effects.

Figure 7: Images without (left) and with (right) colour enhancement

With these improvements in place, the tracking system's performance and robustness were both improved. Moving hands were less easily lost, and background clutter caused less of a distraction.

5 Implementation

The overall aim of the project was to produce an application capable of demonstrating some of the features available in the Medusa distributed multimedia environment. Two such applications were produced; the first was a simple demonstration of the tracking system, and the second used the system as a controller for a puzzle-type game.

5.1 A Tracking and Gesture Recognition Demo

In order to demonstrate the operation of the two algorithms, they were incorporated into a 'toy' application. The application ran in a standard window, displaying the video images being seen by the selected camera. When a hand was held up to the camera, it was located and subsequently tracked until it moved out of view or moved so fast that the tracker lost it. During tracking the hand model was superimposed on the output, so that the tracker's progress could be monitored. Simple gestures, such as the movement of the thumb, were recognised and used to trigger audio samples. Figure 8 shows the tracker in action.

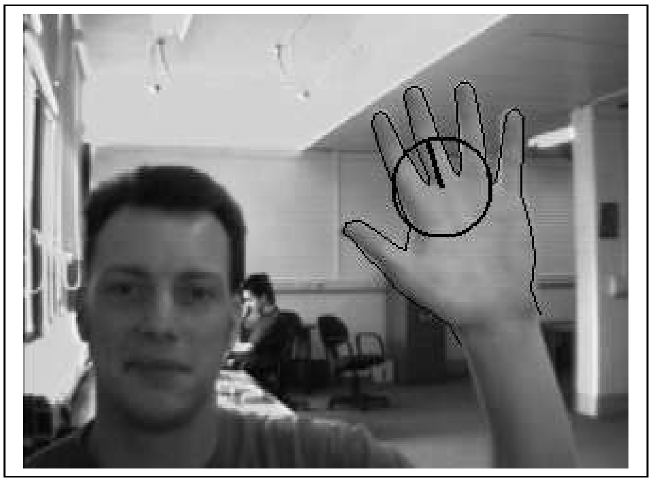

Figure 8: The tracker application in action

Combining the search and tracker modules into a complete, robust application requires a little thought. When no hand is in view, the system should sit idle, watching for potential hand appearances. One way of achieving this is to have the GA search running constantly, until it indicates that it has found a hand in the image. However, the GA search operation is computationally intensive; it would be unsatisfactory to have it running as a background task. A better solution is to have a cheaper, less intelligent algorithm running, whose sole purpose is to constantly monitor the video input, watching for possible hand occurrences. In such an event, the GA search is then activated.

This has been implemented using a simple system which detects large amounts of movement. A hand being held up to the camera is easily detected, but sometimes false alarms occur. However, if no hand is present in the scene then the subsequent GA search will fail, and idle mode will be re-entered. The flow of control for the demo system is shown diagramatically in Figure 9.

The audio cues were triggered when the relevant variation weights reached certain thresholds. These thresholds were discovered by manually monitoring the values whilst certain gestures were performed.

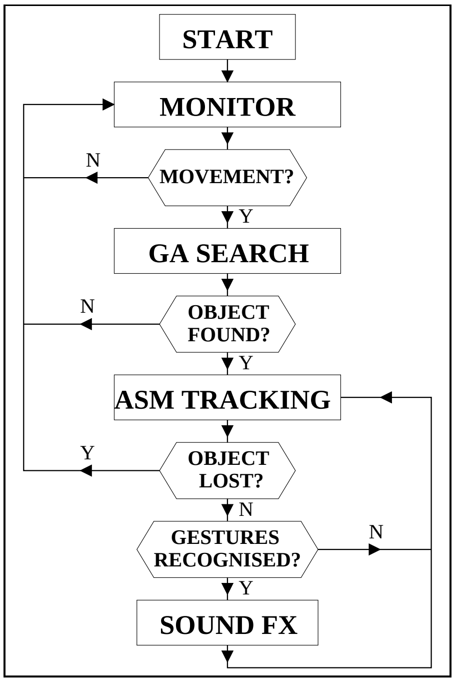

Figure 9: Control flow in the tracker demo

5.2 A 'Gripple' Controller

The tracking system was also successfully incorporated into another application. This was a controller for the puzzle game "Gripple".

The essence of Gripple is that a series of rotation actions must be performed on various discs in order to solve the puzzle, which makes it particularly applicable to hand control, since it is difficult to represent rotation directly using either the mouse or keyboard (other than just having gadgets to operate).

The hand tracker was set up in a similar fashion to the demonstration mentioned above. The x- and y-translation were used to determine which disc was to be rotated, and the rotation value was used to trigger rotation. A feedback window was provided so that the user could see how the tracker was progressing. Users took a little time to get used to the control system, but everyone reported that it was easy to use, although some complained that rapid hand movements were not tracked accurately; one reason for this is that performance is limited by the camera frame rate.

6 Conclusions

It has been shown that Active Shape Models seem to provide a suitable basis for real-time feature tracking, at least in the 2D domain. A major issue is whether or not the technique can extend to 3D. The Point Distribution Model used extends easily into three dimensions, but the limitation in this case is how to perform tracking with only 2D video sequences for input. Possible routes are via stereo vision or shading information.

Acknowledgements

The work detailed in this paper was undertaken during a student placement at Olivetti Research Limited in Cambridge. Thanks are due to all at Olivetti for their help and support, especially to Ferdi Samaria for his unfaltering supervision and useful comments, Rob Walker for the coding of the automatic training utility, Frank Stajano for help with TCL and Tim Glauert and Martin Brown for Medusa expertise.

References

[1] B. Dorner. Hand shape identification and tracking for sign language interpretation. Looking at People Workshop, Chambery, France, 1993.

[2] J.M. Rehg and T. Kanade. Visual tracking of high DOF articulated structures: An application to human hand tracking. In *Proceedings of the European Conference on Computer Vision*, volume II, pages 35–45, Stockholm, Sweden, 1994. Springer-Verlag.

[3] A. Blake, R. Curwen, and A. Zisserman. A framework for spatiotemporal control in the tracking of visual contours. *International Journal of Computer Vision*, 11(2):127–145, 1993.

[4] C. Kervrann and H. Heitz. Robust tracking of stochastic deformable models in long image sequences. In *Proceedings of the IEEE International Conference on Image Processing*, volume III, pages 88–92, Austin, Texas, US, 1994. IEEE Computer Society Press.

[5] T.F. Cootes and C.J. Taylor. Active shape models - 'Smart Snakes'. In *Proceedings of the British Machine Vision Conference*, pages 266–275, Leeds, UK, 1992. Springer-Verlag.

[6] A. Hill, T.F. Cootes, and C.J. Taylor. A generic system for image interpretation using flexible templates. In *Proceedings of the British Machine Vision Conference*, pages 276–285, Guildford, UK, 1993. BMVA Press.

[7] S. Wray, T. Glauert, and A. Hopper. The Medusa applications environment. In *Proceedings of the International Conference on Multimedia Computing and Systems*, Boston MA, US, 1994. (Available as `ftp.cam-orl.co.uk:/pub/docs/ORL/tr.94.3.ps.Z`).

[8] A. Hopper. The Medusa applications environment. Technical Report 94.12 (video), Olivetti Research Laboratory, 1994. (Available as `ftp://ftp.cam-orl.co.uk/pub/docs/ORL/abstracts.html`).