Active-Meshes

D. Molloy, P.F. Whelan[a]

[a]School of Electronic Engineering, Dublin City University, Glasnevin, Dublin 9, Ireland

This paper describes the implementation of an 'active-mesh' that is to be automatically created and configured directly from a single frame of an image sequence. The aim of this approach is to use the derived mesh to perform visual tracking in unconstrained motion environments, allowing movement of the camera, the scene and even the inclusion of background-independent moving objects. The main problem in initializing this mesh comes from the fact that there is little *a priori* information about the scene available. The paper will discuss methods that are currently available for determining the initial position of active contour models within images, also suggesting a method of initializing the active-mesh. The approach is further extended to using multiple meshes and region initialized meshes.

Keywords: Active Meshes, Motion Tracking, Snakes, Active Contour Models

1. Active Contour Models

Active Contour Models are a popular method for tracking 'regions' as features through image sequences. Developed largely by Kass et al.(1987), snakes are active contour models, that demonstrate an example of the generalized technique of matching a deformable model to an image using energy minimization techniques. They can help solve numerous computer vision problems, such as the analysis of dynamic image data, image segmentation and image understanding. The model is described as active since it is always attempting to minimize its energy function, hence showing dynamic behaviour. The shape of the contour determines its internal energy and its external energy is determined by the spatial location of the contour within the image. External forces may be used to attract these contours towards or away from salient image features, such as edges, lines and corners.

The structure of a snake is an ordered set of control points (or snaxels) of the form $V = [v_1, v_2, \ldots, v_n]$ where each snaxel v_i is defined on the finite grid $v_i \in I = \{(x, y) : x, y = 1, 2, \ldots M\}$, allowing every snaxel to have a 2-D coordinate position on the image plane, I. Alterations to the location or shape of the snake are possible by moving the positions of the individual snaxels. The number of snaxels is chosen by selecting an appropriate internal distance h. This value is chosen on an application specific basis, where the coarsness of the fit of the snake to the object is the defining factor.

There are several models of deformable contours that may be used, such as the snake model suggested by Kass et al.(1987) that 'wraps around' image features, or the balloon model introduced by Cohen (1991) that expands to locate desired image features. Staib and Duncan (1992) examine other methods such as elliptic Fourier decomposition for objects with shape irregularities. Active contours are useful for tracking particular objects within an image sequence, especially when knowledge is provided *a priori* about those objects. This paper examines the creation of active-meshes that track image features in a similar way to snakes, but require little *a priori* knowledge about the scene.

2. Why Active-Meshes?

Many methods for developing active contour models have emerged in recent years, since the introduction of snakes by Kass et al.(1987), with many applications including Staib and Duncan (1992) and Leymarie and Levine (1993). However, in most of these applications it is assumed that the initial position of the snake is relatively close to the desired solution, often initialized by

a human operator (such as the method of Etoh et al. (1993), and Cohen (1991)). While this might be a suitable assumption in some cases, automated initialization is usually required.

The correct initialization of the snake is a difficult problem that has a significant impact on the snake's final location. If the snake's initial position is spatially far from the desired solution, it is quite common for the snake to become trapped in local energy minima, due to irrelevant edge information or noise. The snake is also limited in spatial movement since the snake's own potential energy prevents the snake from moving far from its current position. If the snake is placed close to an intended contour, its energy minimization behaviour will force the correct solution. Snakes do not attempt to solve the problem of detecting prominent image contours, but rely on other mechanisms to place the snake near the desired contour. Some of these mechanisms include: (i) The *Hough transform* which is commonly used for the extraction of the initial estimates of the contour position for rigid objects. These rigid templates cannot account for deformations that may occur, thus a rigid template chosen *a priori* cannot produce satisfactory results in all cases. (ii) *Short snakes* may be initialized at strong edges and allowed to expand and even overlap until the entire boundary is covered. (iii) *Randomly initialized snakes* may be placed on the image, until a suitable solution is found, however, this is rarely practical. Wang and Lee (1994) suggested the use of a quadrilateral structured mesh, which they use in video coding applications. Their energy minimisation approach allows non-uniform area samples of image sequences to predictively describe that image sequence. Our paper uses meshes for tracking, with strong feature points used as the vertices of the mesh, providing clear localisation for tracking and allowing motion description similar to that of feature matching.

3. Active-Mesh Algorithm

The trajectories derived from locating similar feature points on an object, through multiple frames are popular as they are relatively easy to extract. The generation of motion trajectories from a sequence of images typically involves the detection of tokens in each frame, and the correspondence of these tokens from one frame to another. These tokens, which may include edges, corners, interest points, and regions need to be distinct enough for detection and stable enough to be found in each frame. Once the tokens have been correctly matched the real-motion of the selected points in the image plane are available as vectors.

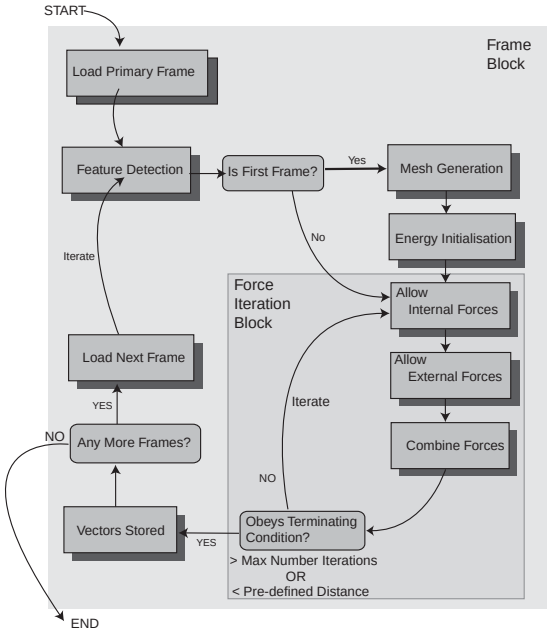

Figure 1. An outline of the active-mesh algorithm.

The active-mesh algorithm allows constrained feature matching to take place on a frame-by-frame basis in an image sequence. Fig. 1 illustrates the overall operation of the active-mesh algorithm. There are two main components to this algorithm, the *Frame Block* and the *Force Iteration Block*. Once the initial image has been loaded, the SUSAN grey-level corner detector of Smith (1992) is used to determine the location of

the mesh nodes.

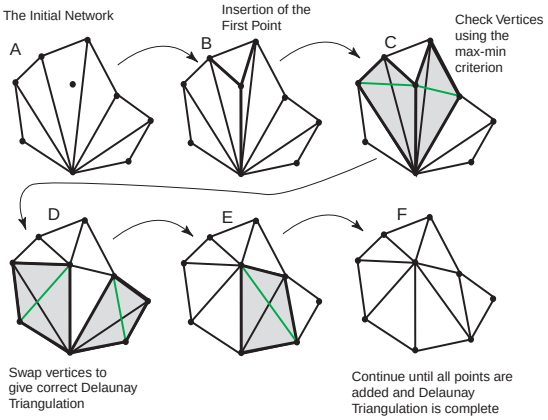

Figure 2. The creation of the Delaunay triangulation using an Incremental Algorithm.

3.1. Mesh Generation

The interconnecting physical structure of the mesh is then created using a modified iterative Delaunay triangulation algorithm (Shewchuk, 1997). Given a set of data points, the Delaunay triangulation (Fig. 2) produces a set of lines connecting each point to its natural neighbours. A Delaunay triangulation is desirable for approximation applications because of its general property that the majority of the triangles are almost equiangular and also because there is a unique triangulation for a given set of points. Unlike many other algorithms for determining the Delaunay triangulation, the incremental algorithm has the prime advantage of keeping the triangular network as a Delaunay triangular network during the actual triangulation process. In the case of the active-mesh, cases can arise where mesh nodes become occluded, leave the scene, or become unreliable. In this case, or in the case where new nodes appear, the incremental Delaunay triangulation algorithm allows the addition or subtraction of mesh nodes (Shewchuk, 1997). The active-mesh algorithm uses this property and structured force models to allow mesh nodes to be removed or added dynamically during frame iterations, therefore enabling the active-mesh to be adaptive (see Sec. 4). Fig. 3, demonstrates an actual example test scene, in which this initialization has been performed. It can be seen from this test image that the feature corner points are detected accurately and the mesh is well constructed by the Delaunay algorithm.

The algorithm initializes the mesh using feature points (SUSAN corners) extracted from the initial image frame to provide the initial node locations of the mesh. Once the mesh is generated it is initially stationary, as no internal or external forces are being applied. The mesh will remain stationary until a change in the underlying image structure occurs.

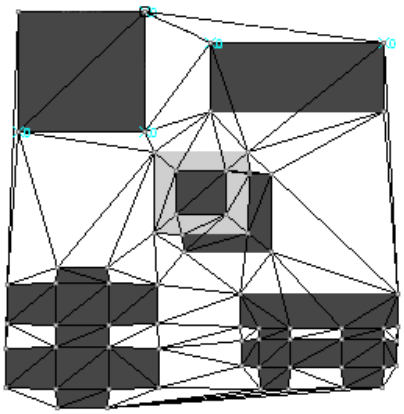

Figure 3. The mesh created using the modified Delaunay triangulation using the detected SUSAN corners as the node points.

3.2. The Internal Forces

The mesh-lines have elastic properties, so that the mesh can deform when required over a number of force block iterations. This allows tracking to be performed, even when global deformation of the scene or local deformation of the objects is occurring. In our model the forces applied to a

node are proportional to the difference in length between the current length of the interconnecting line and a reference length. This is quite distinct from the standard snake model where the forces are proportional to the distance between a snaxel and its neighbours. The elastic properties give the mesh its flexibility, while the rigid properties give the mesh its structure. The rigid properties of the mesh lines cause the lines to attempt to return to their reference length. This reference length is permitted to expand or contract slowly over a number of iterations (in the *Force Iteration Block*), influenced by the elastic properties of lines. In other words, if the mesh is stretched by a number of consistent external forces for a significant number of iterations then the mesh will slowly assume a new default shape. This default shape is now the rigid shape of the mesh and will remain so, until similar forcing conditions arise. The mesh lines have a current length (L_{cur}) and a set length (L_{set}), so for each line in the mesh the generated line force is:

$$\overline{F}_{Line} = L_{cur}(x)\beta_L\overline{i}_x + L_{cur}(y)\beta_L\overline{i}_y, \quad (1)$$

where $\beta_L = (\frac{L_{set}-L_{cur}}{\alpha_L L_{cur}})$, $L_{cur}(x)$ and $L_{cur}(y)$ represent the x and y components of the current mesh line lengths, $\overline{i}$ is a unit vector with the same axis as the image and α_L is a user-defined factor that limits the effect of the forces. The internal forces are determined by the current-length of each individual mesh line in comparison to the set-length of that mesh line. The forces applied to each of the two mesh line nodes n_1 and n_2 are:

$$\overline{F}_{n_1} = -\overline{F}_{Line} \; and \; \overline{F}_{n_2} = \overline{F}_{Line} \quad (2)$$

At each iteration:

$$L_{set} = L_{set} + \alpha_I(L_{cur} - L_{set}) \quad (3)$$

where α_I is a user-defined factor that limits the change in the line length. Fig. 4 illustrates the effect of deformation in the mesh. In Fig. 4(a) a force is applied to both ends of the mesh line. The mesh line expands (as L_{cur}), but eventually contracts back to the initial length L_{set}. In Fig. 4(b) deformation is allowed, so when a force is applied to both ends of the mesh line for a length of time δt the L_{set} begins to slowly assume this new length L_{cur}. When applied to all the lines in the mesh, it has the effect of allowing deformations in the shape of the mesh.

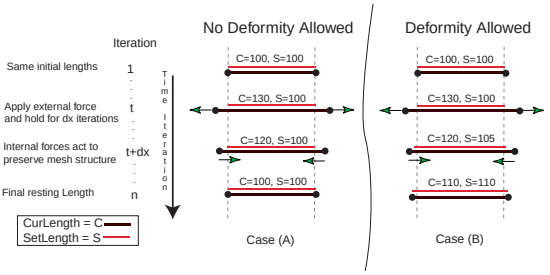

Figure 4. Example of a mesh line deformity in (a) a rigid mesh (b) a deformable mesh.

3.3. The External Forces

External forces are applied to the mesh nodes independent of the mesh lines and are derived from the image data. These image forces pull the mesh nodes towards suitable feature match points that are found within the circular image search space of each mesh node. If a suitable match feature appears within the circular search space, then the node is pulled towards that feature point by a force magnitude determined by the suitability of the match feature. This in turn pulls the connected mesh nodes (due to the internal forces of the interconnecting mesh lines) in the direction of the new feature.

The best match corner is found by comparing the 3×3 area surrounding the current node n_0 with the 3×3 area surrounding the possible match corners c_n detected within the circular search space of radius r.

$\forall c_n$ (with coordinates (x_{c_n}, y_{c_n})), where the Euclidean distance to the current mesh node n_0 (with coordinates (x_{n_0}, y_{n_0})), d, is less than the search space radius, r, the total intensity difference is:

$$I_T = \sum_{i,j=-1}^{1} |I(x_{n_0}+i, y_{n_0}+j) - I(x_{c_n}+i, y_{c_n}+j)| \quad (4)$$

The corner point c_n is chosen that minimizes the value of I_T in the range 0 to 2295 (i.e. 255×9). Based on this intensity difference, a match strength is established:

$$S_M = 1 - (\frac{I_T}{9 \times 255}) \tag{5}$$

The value of I_T is averaged over 9 pixels and normalized over the intensity range 0 to 255, therefore $S_M = 1$ for the best possible match and 0 for the worst possible match. For a single node, the external force is:

$$\overline{F}_{ext} = \beta_{ext} d(x) \overline{i}_x + \beta_{ext} d(y) \overline{i}_y, \tag{6}$$

where,

$$\beta_{ext} = \alpha_E S_M (\frac{r - d}{r}) \tag{7}$$

and where α_E is a user-defined factor to allow the external forces to have a larger or smaller effect on the mesh. The last term weights the strength of the force as weaker the larger the distance the feature is from the examined node.

3.4. Force Combination

To allow for the complexities involved in dealing with active-meshes as opposed to the more simple contours, the active-mesh algorithm must allow for varying numbers of mesh line connections to each node. It must also provide an algorithm for dealing with the numerous forces that may be applied at each node.

It was determined experimentally that the closer a connected node was, the more likely it is that this node is present on the same real-world object. So examining center node n_0 with connected nodes $n_1, n_2, \ldots, n_N$ the combination is of the form (using β_i as a weighting factor):

$$\beta_i = 1 - \frac{D_i}{\sum_{i=1}^{N} D_i}, \tag{8}$$

$$\overline{F}_0 = \sum_{i=1}^{N} \beta_i \overline{F}_i, \tag{9}$$

where $\overline{F}_0$ is the force on the centre node, and D_i is the Euclidean distance between n_0 and n_i. The larger the distance of the node applying the force from the current node, the smaller the effect it has on the movement of the current node. The forces applied from the surrounding nodes may be caused by those surrounding nodes being pulled towards or away from salient image features. Fig. 5, illustrates multiple forces being applied to a single node, the combination of which is performed using the force strength and the distance of the forcing nodes from the center node as described in Eqn. 8 and Eqn. 9.

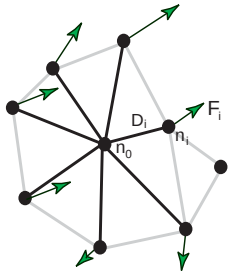

Figure 5. Multiple forces being applied to a single node.

3.5. Next Frame

Once the force combination has taken place the mesh structure is examined to see if it obeys the terminating condition (Fig. 1). This condition is usually defined as: "if no mesh-node in the mesh has moved more than 1 pixel since the last force iteration, then terminate the *Force Iteration Block*".

Once this condition is true, the vectors are stored, representing the distance and direction that each node has traveled since the last image frame. The reference length of each mesh line is updated to the current length ($L_{set} = L_{cur}$). The next frame image is loaded, feature detection takes place and the *Force Iteration Block* is once again entered.

4. The Adaptive Active-Mesh

The adaptive mesh algorithm is very similar to the general active mesh algorithm but with a few extensions. The concept behind the adaptive mesh is clear and there are two main choices in the adaption:

Drop features: If the mesh is tracking a set of tokens for several frames, it should drop the tokens that are weakening the overall matching by being inconsistently detected over image frames. So the following cases should cause selected tokens to be dropped: (i) Weak features that are inconsistently detected. (ii) Features that go out of scope. Features that are tracked may leave the image view and can, if desired, be deleted. (iii) Features that become occluded can be deleted.

Add features: If the mesh is tracking a set of features for several frames, it can also add features to the mesh. The features to be added might be caused by: (i) Dis-occlusion of image features, where features might appear in the image frame due to an object moving and revealing previously occluded features. (ii) New frame edge features. Features will likely be detected at the edges of the image frame when the observer view moves to the left or right of the original view. (iii) New moving object. The entrance of a new independently moving object may cause new features to appear. Key to the concept of the active-mesh was its ability to deal with changes in structure as the image sequence progressed. The concept of using the iterative Delaunay triangulation was for this very reason.

5. Multiple Meshes

The active-mesh algorithm has outlined the structure to allow multiple meshes to exist in the same image sequence. The determination of the locations of the individual meshes can be performed in several ways. The primary and most obvious way to determine multiple meshes is to examine the motion vectors of a large mesh and sub-divide the mesh into child meshes that are subsets of the larger mesh. If required, this child mesh can then exist with no relationship to the parent mesh, as the nodes can be removed that correspond to the child meshes and the exist independently. The identification of an independent similarly moving cluster of mesh nodes was implemented and used to identify these child meshes.

The facility to have multiple meshes was added to the algorithm by adding an array of meshes. Once initialization has been performed, the individual meshes are threads that operate in parallel, with no transfer of information between individual threads. Currently, the initialization of the multiple active-meshes is based on a calculated region mask.

6. Discussion

Several different active-mesh approaches have been described - active-mesh, adaptive active-meshes and multiple active-meshes. The active-mesh and adaptive active-mesh approaches are very suitable for unconstrained motion tracking, whereas the multiple region based approach is very suitable for fixed camera applications, as well as moving camera applications where multiple objects are involved.

Some of the advantages with these methods include: (i) The initialization of the mesh(es) is automatic the algorithm parameters require little more than tuning. The tuning parameters allow the user to define the rigidity of the mesh and the strength of external forces. The default values were suitable for most of the applications examined. (ii) It is based strongly on 2-D image features, which are shown to be less likely to suffer from local effects, such as the aperture problem. These features have also been found to be accurately defined and consistent between image frames. This algorithm takes the advantages of lower-level feature matching and provides a higher-level framework. (iii) The energy-based approaches of active contours have gained acceptance as a suitable framework for motion tracking, with this approach allowing a more structured model for dealing with irregularly shaped objects than snakes or balloons. (iv) There are few assumptions made about the scene and no *a priori* information is required. (v) The system is modular. (vi) The algorithm can deal with affine transforms of the object as well as deformations

within the tracked object. (vii) The algorithm currently operates on a 2-frame basis. Multiple image frames and Kalman filtering would certainly improve the quality of the tracking over multiple frames. (viii) The calculated motion vectors lengths between frames can be quite large - typically of the order of 20 pixels, which is very impressive in comparison to the traditional approach of optical flow.

Primarily, the number of mesh-nodes and the number of features in the search space of each mesh node define the complexity of the active-mesh algorithm. The complexity could be defined by $O_1(n) \times O_2(m)$ where n is the number of mesh nodes, m is the average number of features in each search space, O_1 is the mesh update operation and O_2 is the feature-matching algorithm that is performed for each of the m features in each nodes circular search space.

7. Applications

A specialised software application was developed to implement the active-mesh algorithms. Fig. 6 demonstrates one possible application of the multiple mesh algorithm[1]. In this scene the camera is fixed, so temporal background subtraction is used to isolate the current image from a reference image. This identifies the initial multiple regions. Once the meshes are initialised on these regions they follow the feature points that were associated with those regions, so the individual meshes use the best match feature points to continue to track the objects, avoiding the static frame feature points.

Fig. 7 displays the results from the castle sequence. The mesh tracks the castle as it moves to the right and down in the image over 12 frames. The paths displayed represent this motion at the mesh node points over these 12 frames. Some of the points present on the mesh leave the image area but still remain part of the mesh. As they are not attracted strongly to specific features, they have little effect on the movement of the mesh, rather they preserve their internal shape, as they would if occlusion occurred. This is a suitable property in this application as the motion direction could once again change and they could re-enter the image area. If desired the adaptive mesh properties may be used to simply remove nodes that leave the image area and also to add new features that appear as mesh nodes.

Currently the algorithm is implemented entirely in Java for development purposes, taking about 0.5 seconds per image frame to process a 500-node mesh on a Pentium II 200MHz processor.

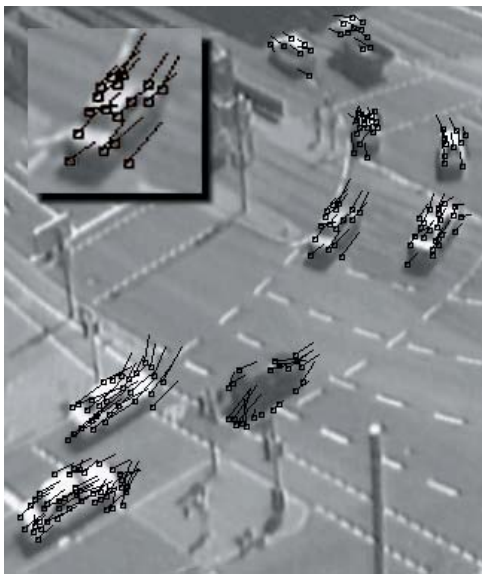

Figure 6. Results of multiple meshes, used on 2 frames of the road sequence.

8. Conclusions

The approach outlined in this paper illustrates the use of a self-initializing active-mesh. It has primary application in scenes where little *a priori* knowledge is available, or where unknown motion events can occur in subsequent image frames. The initialization of the mesh location is based on an incremental Delaunay triangulation, so that

[1]Images obtained from the VASC Image Database http://www.ius.cs.cmu.edu/idb/

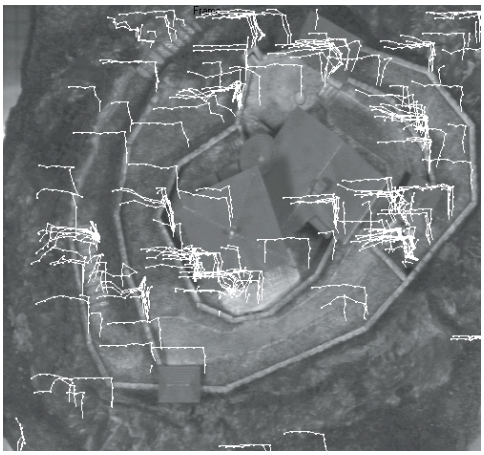

Figure 7. Results of adaptive mesh, used on 12 frames of the castle sequence.

node points may be added and removed dynamically as they appear and disappear on a frame-by-frame basis. The force equations that were used were specifically designed for the initialization technique developed. The technique provides promising tracking results, providing information about the 'real' motion in the scene, which can be difficult for techniques such as optical flow. This method is currently being applied to 3-D scene reconstruction.

For further information on this work, additional results, moving result sequences and for an interactive Java applet see: http://www.eeng.dcu.ie/~molloyd/meshes/

9. Acknowledgements

We would like to thank the reviewers and Alex Drimbarean for their suggestions and contributions to this paper.

REFERENCES

1. Kass, M., Witkin, A., Terzopoulos,D., 1987. Snakes: Active Contour Models *Proceedings of the 1st International Conference on Computer Vision* , 259–268.
2. Cohen, L.D., 1991. NOTE On Active Contour Models and Balloons. *Computer Vision Graphics Image Processing: Image Understanding* , 53, **2**, 211–218.
3. Staib, L.H., Duncan, J.S., 1992. Boundary Finding with Parametrically Deformable Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14, **11**, 1061–1075.
4. Leymarie, F., Levine, D., 1993. Tracking Deformable Objects in the Plane Using an Active Contour Model. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15, **6**, 617–633.
5. Etoh, M., Shirai, Y., Asada, M., 1993. Active Contour Extraction Based on Region Descriptions Obtained from Clustering, *Systems and Computers in Japan*, 24, **11**, 55–65.
6. Smith, S.M., 1992. A New Class of Corner Finder, *British Machine Vision Conference 1992*, Leeds, 139–148.
7. Wang, H., Lee, Y., 1994. Active mesh - A Feature Seeking and Tracking Image Sequence Representation Scheme, *IEEE Transactions on Image Processing*, 3, **5**, 611–624.
8. Shewchuk, J.R., 1997. Delaunay Refinement Mesh Generation, *Ph.D. Thesis, Carnegie Mellon University*, CMU-CS-97-13.