Tracking Objects Using Image Disparities

Alistair J Bray
School of Cognitive and Computing Sciences
University of Sussex
Janet: alib@cogs.susx.ac.uk

A method and results are presented for a system that finds and tracks known polyhedral objects in 3-space, given a sequence of grey-level images. The object is located in the first frame using **model-based search**. *Image features are then tracked into the next frame using* **optic flow** *techniques, and their disparities are used to* **invert the perspective transform**. *The system can recognise when it is getting lost during the tracking stage. It recaptures its position through another model search that uses the reduced disparity information, and a 'best guess' at position, to constrain the size of the search space. To do this the system integrates 3 established algorithms in a novel way: model matching is done using modified Goad-search [1, 2]; edgelets are tracked between images using a flow algorithm such as Barnard & Thomson's [3]; and the perspective transform is inverted using Lowe's formulation of the projection equations [4, 5].*

A model-based system that uses 2D features encounters a very large search space in the matching process, since such features are *viewpoint dependent*; however, such 2D features are generally much simpler to compute than the alternative – 3D features. This work demonstrates that when tracking an object through successive images it is possible to avoid both a combinatorial explosion of the search space, and the problems of complex feature construction, by *exploiting knowledge obtained from previous frames.*

There are two possibilities for exploiting the known position of an object in a frame: it can allow the use of image-based techniques in tracking the object into the next frame, or else it can constrain a model-based search for the object in the next frame. The former is fast but can fail; the latter allows recovery from such failure. This system uses both methods; it is therefore integrating model-based search with image-based tracking to exploit the benefits of both.

SYSTEM OVERVIEW

The system finds the object in the first frame. It then tracks the object through successive frames until it starts to fail, at which point it uses the partial information available from the tracking process to recover lost position and continue as before. The system divides naturally into 3 stages of processing:

1. The **First Frame** position is found using a model-based search method – a modification of Goad's algorithm [1, 2] – followed by Lowe's convergence [4, 5]. The Goad-search performs a match between model features and image features to locate the object in a particular frame . It calculates two things: a *set of correspondences*, or an *interpretation*, between image and model features for a particular frame and a *locus of viewing positions* for which this set is consistent [1] . Lowe's convergence method takes this derived correspondence set, and a rough estimation of the location of the object computed from the viewing locus, and performs two functions. It verifies that the correspondence set is globally consistent, and it provides a more accurate estimation of object position by making maximal use of the information in the correspondence set.

2. The **Tracking Stage** takes as input a precise position for the object in the first frame (such as is found in Stage 1). By projecting the model at this position it can predict features in the image and track them into the next frame using image-based techniques. By doing this it is able to deduce the correspondence set for the next frame without resorting to the expense of a model search and segmentation. Given the correspondence set it performs a convergence from its position in the first frame to its position in the next (as above). This process repeats itself, tracking the object through successive images, until it decides that it is unable to locate the object in the next frame reliably. It then passes control to Stage 3.

3. The **Recovery Stage** takes over if tracking gets lost. It takes a reduced subset of the correspondence set and a very rough estimation of object position (such as is available when tracking fails). It conducts a Goad-search as outlined in Stage 1. However the search is centred on those viewpoints that correspond to the position estimate, and is constrained by the degraded correspondence set. As with Stage 1, it yields a reduced locus of viewpoints and a fuller correspondence set, which are then used for convergence to give the most accurate object position possible.

The flow of control between these stages is therefore very simple. Stage 1 finds the object in the first frame of a sequence. Stage 2 then tracks the object through subsequent frames for as long as it is able, using image-based methods to avoid search. Stage 3 is entered only

[1] A view position corresponds to a point on the viewsphere and constrains object orientation to within one degree of freedom

if previous tracking breaks down; it re-establishes lost position via a constrained model-search and then passes control back to Stage 2 to resume tracking.

ALGORITHM

In the system implemented each of the above stages uses adaptions of established algorithms.

First Frame

The object is detected in the first frame using a model search that is a development of Goad's algorithm. The model for the object class is defined as a set of 3D line segments. The image is processed for 2D line segments. The viewsphere is quantised into a set of viewpatches, and the interpretation tree of potential matches between image and model lines is then searched over the set of viewpatches. The tree is pruned using local constraints that have been pre-compiled off-line. As a branch of the tree is expanded the possible position of the object is restricted. Corresponding to each node in the tree there is a locus of viewpatches over which the above branch is consistent. At a leaf-node this locus defines a *rough* estimation of the orientation of the object for the first frame. A branch of the tree that reaches a leaf-node also defines a possible correspondence set (or interpretation) between model and image features, where each node in the branch constitutes a member of the set.

The local constraints used for pruning the tree correspond to the *Angle Constraint* and the *Direction Constraint* described by Grimson [6] and illustrated in Figure 1. These constraints can cope with the object being at any scale in the image (or any distance from the camera). They are also insensitive to errors resulting from image segments being of reduced size due to occlusion, poor lighting or poor segmentation. This *scale independence* and *segmentation independence* is due to the fact that both pruning constraints used are angular. The Angle Constraint puts bounds on the permissible angle between two image segments, and the Direction Constraint puts bounds on the direction of a vector connecting two image segments. Measures of distance between segments are totally ignored. The algorithm still relies upon a relatively good segmentation of the image however, in that failure to detect lines that should be visible leads to unacceptable expansion of the search space.

Given the partial orientation, constrained to within one degree of freedom, it is simple to determine rough estimates for the final orientation and the three translation parameters. The partial orientation allows a 2D *template* to be generated from the model. This template can be rotated, translated and scaled in 2D to match against the image, matching being guided by the correspondence set. The 2D rotation can be composed with the partial orientation to remove the last degree of freedom; the 2D translation determines 2 of the 3D translation parameters, and the third (the depth parameter) can be estimated from the 2D scaling factor.

Given the correspondence set and these estimates for object position the model converges to a more exact position by inverting the perspective transform. The method

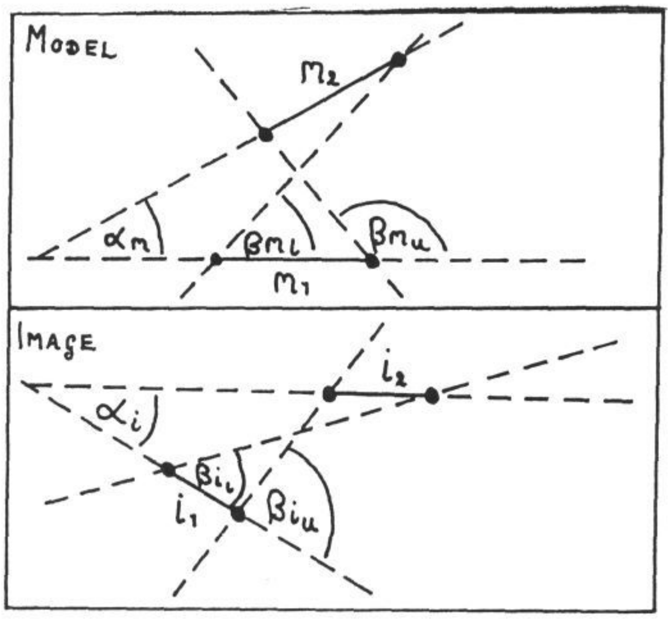

Figure 1: **The Pruning Constraints.** *In the ideal case the Angle Constraint states that* $\alpha_m = \alpha_i$ *and the Direction Constraint states that* $\beta_{m_l} \leq \beta_{i_l}$ *and* $\beta_{m_u} \geq \beta_{i_u}$

used is that of Lowe. The model is projected into the image at the estimated position to give a set of image lines. Each of these is compared to the actual image segment to which it corresponds (deduced from the correspondence set) to give a set of disparities. Each match yields two disparities – the distance of the ends of the projected model lines from the *infinite line* defined by the image segment. These disparities are used as the error terms when inverting the transform: an adjustment is calculated for the object rotation and translation which, when composed with the initial position estimate, provides a new estimate of object position such that the disparities are reduced. This is an iterative process that halts when the adjustments are zero, and the model has converged to a final position. Lowe's formulation of the projection equations results in fast convergence. The choice of error term means that the algorithm is ignoring information concerning the length of the image segments found, and is only using the transverse position. This leads to an accurate solution given sufficient non-parallel image segments.

Tracking

To find the object again in the next frame it is not necessary to repeat the expensive Goad-search since 2D line segments can be tracked directly, thereby maintaining the correspondence with model lines. To take advantage of surface texture and other contextual information, line segments are tracked via a flow field computed between the consecutive frames using Barnard & Thomson's algorithm applied to edgelets [3]. Alternative algorithms have been experimented with for computing the flow field: standard grey-level correlation techniques were used to compute the flow for the result sequence shown in Figure 5 [2] . Once the correspondence set has

[2]Since only the transverse component of flow is required for feature-tracking, simple techniques provide satisfactory results.

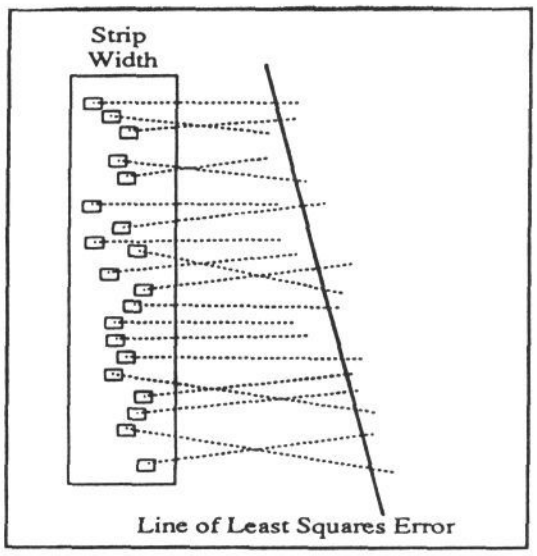

Figure 2: **Mapping To Infinite Image Lines.** *Edgelets found close to the projection of a model segment are tracked into the next frame. They are used to compute the infinite image line*

been computed in this way, the model can converge from its original position to a new position in the next frame, as described above.

When tracking features using flow, model segments are mapped onto infinite lines and images need be segmented no higher than edgelets. Firstly, 2D disparities are calculated for all edgelets moving from one frame to the next using the chosen flow algorithm. All model segments are projected in the first frame at the known position , and a strip of variant width centred around the 2D segment is overlaid on the edge-image for this frame. All edgelets falling within the strip are considered constituent of the model segment; their new positions in the second frame are accessed from the flow field, and the *line of least squares error* passing through them is computed. If the corresponding correlation co-efficient is sufficiently high then that model line will be taken as both visible and successfully mapped. This simple tracking technique is illustrated in Figure 2.

As already stated, tracking the projection of model lines means that it is not necessary to perform a full segmentation of the image: edgelets need not be grouped into line segments. The technique provides a mapping between model lines and *infinite image lines* for a given frame; this is precisely what is required for convergence. Since the mapping is to *infinite* lines, the aperture problem that commonly confuses edgelet tracking algorithms is neatly side-stepped – it is only important that edgelets are mapped onto the correct line, the position along the line is irrelevant. The method of thresholding on correlation coefficient makes the algorithm robust to both general and self-occlusion in that occluded segments can be easily filtered out; the algorithm can also cope with self-occlusion by predicting which model features should be visible at the projected position.

This process of tracking features followed by convergence is repeated for subsequent pairs of images until it fails.

More sophisticated algorithms [7, 8, 9] have not been tested

Recovery

When tracking starts to err, which can be determined either by examination of parameters in the convergence stage or by the number of image features being successfully tracked, a constrained Goad-search is applied. The system backtracks to the last frame in which object position is known with certainty (usually the last frame processed) and the next frame is then segmented. A subset of the correspondence set is computed for the segmented frame using the most successful feature mappings from the feature tracking stage: the image segment in the segmented frame that maps best onto the infinite line defined by the feature mapping is considered to correspond to the model line defined by the feature mapping. This subset is then used, along with the last position estimate, to seed a Goad-search. The partial correspondence set defines a sub-branch of the search tree, and the search can operate on a reduced viewpatch locus that is centred on the patch corresponding to the position estimate, and of a size determined by possible interframe motion. The sub-branch is checked for consistency – if it fails a new branch is sought – and the locus it defines further reduces the search locus. The model search then continues from the bottom node of the branch.

Since the search tree branches highly at the top and substantially less at the bottom the size of the search space is greatly constrained. If the recovery stage is entered as soon as tracking begins to fail the search will be very fast, and the worst case of unconstrained search can be avoided. The result is a fast computation of the full correspondence set, and a more accurate estimation of object position. These are used, as before, as input to the convergence routine which provides a final estimation of object location. In fact it is the complete correspondence set that is most significant, since the last known object position is usually accurate enough to be used in the convergence routine.

RESULTS

The following results demonstrate the system tracking a 3-point plug through 2 sequences of images.

- **Figure 3** shows a wire-frame drawing of the model for a 3-point plug, the edges extracted from the first frame of Sequence 1 using the Canny operator [10] (thresholded on edge-strength), the segmentation of the first frame, and the edgelet flow chart between the first and second frames (produced using Barnard & Thomsons algorithm).

- **Figure 4** shows the model overlaid in Sequence 1 – a sequence of 5 frames [Resolution 256x256]. The first image shows the model position in the first frame after the Goad-search but *before* convergence. The next five show the model's final position *after* convergence for all five frames of the sequence.

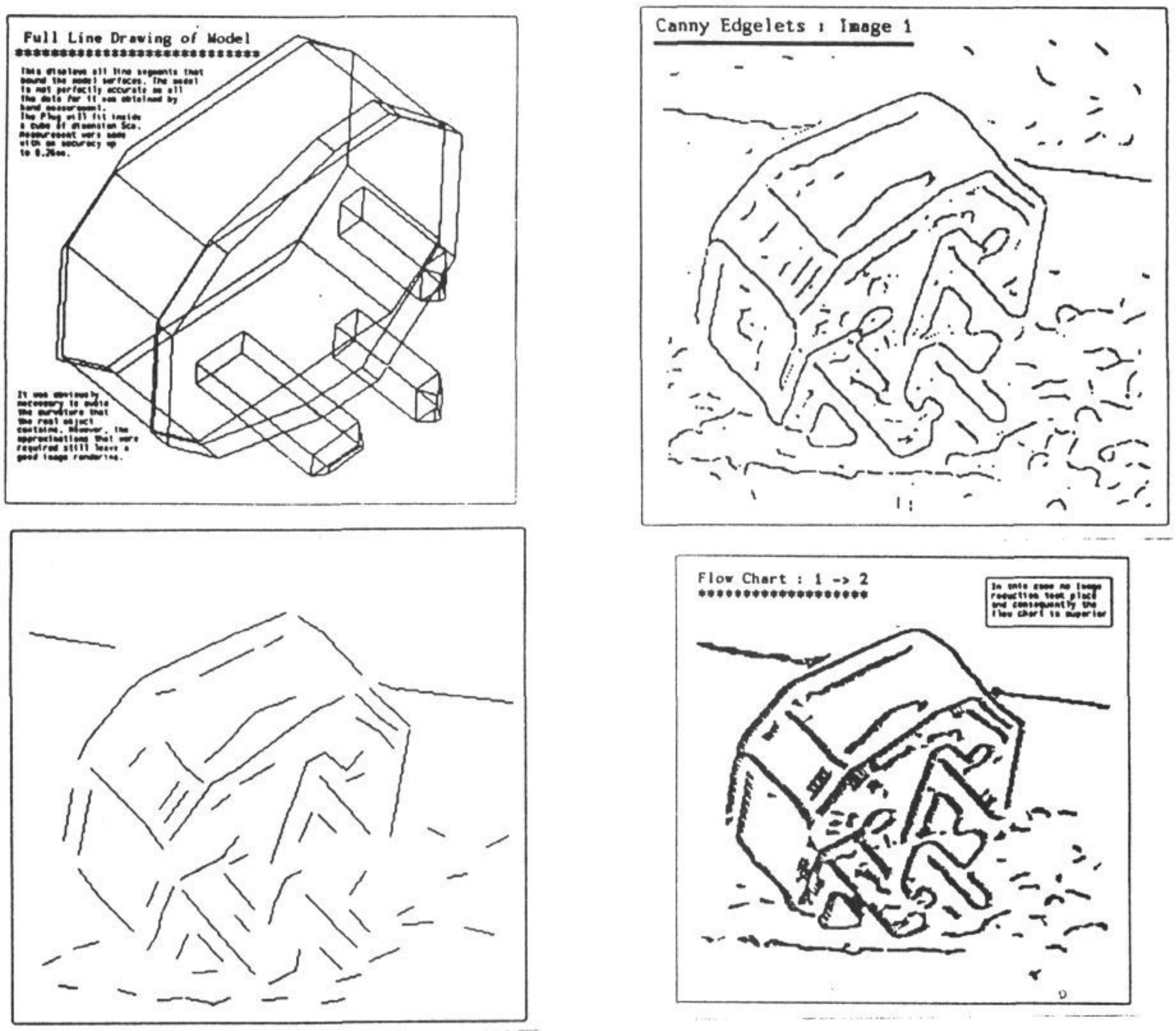
Full Line Drawing of Model
This displays all line segments that bound the model surfaces. The model is not perfectly accurate as all the data for it was obtained by hand measurement.
The Plug will fit inside a cube of dimension 5cm.
Measurement were made with an accuracy up to 0.25mm.
It was obviously necessary to avoid the curvature that the real object contains. However, the approximations that were required still leave a good image rendering.
Canny Edgelets : Image 1
Flow Chart : 1 -> 2
In this case no image reduction took place and consequently the flow chart is superior

Figure 3: **Model, Edges, Lines & Flow**

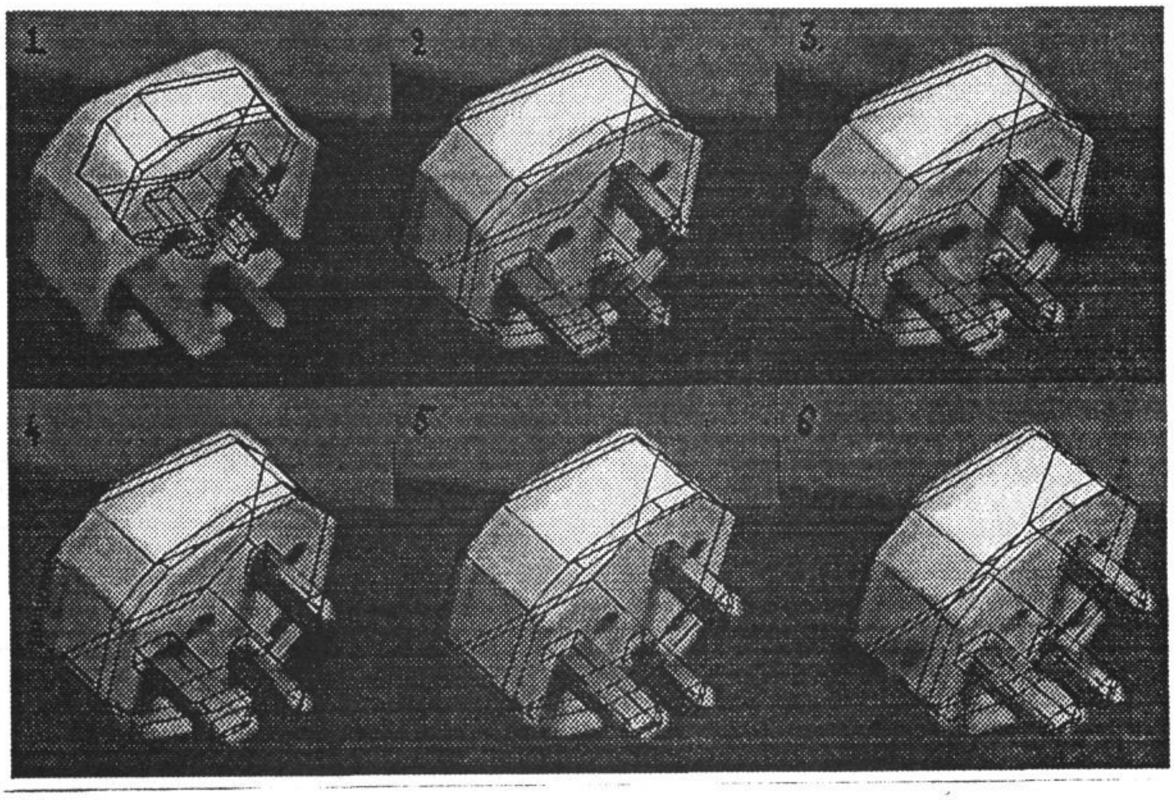

Figure 4: **Sequence 1**

- **Figure 5** shows the model overlaid in Sequence 2 – a sequence of 16 frames [Resolution 128x171] of the plug swinging on a string. The images contain poor quality edge information and the first image had to be *hand-segmented* for the Goad algorithm to work satisfactorily. The model is overlaid at its final position throughout the 16 frames. The flow was computed for this sequence using a correlation algorithm similar to the *Sum of Squared Differences* method [11].

The first sequence demonstrates the system working without help; the second shows that, given hand-segmentation, the tracking process operates effectively on low resolution data. Neither sequence demonstrates the recovery stage.

SUMMARY

- A system has been designed and implemented that **integrates image and model based techniques** in tracking polyhedral objects in 3-space. It is potentially both fast and robust. The image-based processing gives it speed while the model-based processing allows it to find the object in the first frame, and to recover if it starts getting lost. In both types of processing, information concerning the current object position is used – either to constrain search or to track features.
- The system **avoids many segmentation problems**. For the most part segmentation of the image into line segments is unnecessary. When it is necessary, only angular relationships between segments are exploited in model matching, and only the transverse position of segments is exploited in convergence.
- The system manages to **side-step the aperture problem** when calculating flow. When tracking features the projected model segments are being mapped onto infinite image lines in the next frame and hence only the transverse component of the flow vector is neeeded to compute this mapping.
- The system has the ability to cope with a degree of **general occlusion**: the constraints used in the Goad-search are insensitive to segment length; only features that are visible will be successfully tracked; and the convergence routine uses error terms that are also insensitive to segment length. The system (esp. the Goad-search) *cannot* cope with many expected features being totally absent from the image.
- **Results** have demonstrated that the system is capable of tracking polyhedral objects in poor quality low-resolution image sequences. The model-based module requires better quality images, since errors in detecting line-features result in an unacceptable expansion of the search space.
- **Further work** would be worthwhile in the following areas:
  - Incorporating information concerning object motion into the predictive modules
  - Adapting the model-based algorithm to work on a variety of image features other than line segments. This would allow the system to work in a variety of image domains and with a more flexible modelling system (eg. for curved objects).
  - Finding alternative solutions to the correspondence problem on these different types of features

ACKNOWLEDGEMENTS

Thanks go to Dave Hogg and Vaclav Hlavac for help with this research, and their comments on the paper.

References

[1] **C. Goad**. "Special purpose automatic programming for 3d model-based vision". *Proc. Image Understanding Workshop, Virginia, USA*, pages 94–104, 1983.

[2] **C. Goad**. "Fast 3d model-based vision". In A. P. Pentland, editor, *From Pixels to Predicates*, pages 371–391. 1984.

[3] **S. T. Barnard and W. B. Thompson**. "Disparity analysis of images". *Technical Report 79-1, CS Dept., University of Minnesota*, 1979.

[4] **D. G. Lowe**. *Perceptual Organisation and Visual Recognition*. Boston: Kluwer, 1985.

[5] **D. G. Lowe**. "Three-dimensional object recognition from two-dimensional images". *Artificial Intelligence*, 31(3), 1987.

[6] **W. E. L. Grimson and T. Lozano-Perez**. "Model-based recognition and localisation from sparse range or tactile data". *International Journal of Robotics Research*, 3(3):3–35, 1984.

[7] **G. L. Scott**. "A single method for extracting both point and edge motions from a pair of images". *Proc Alvey Vision Conference, Bristol 86*, 1986.

[8] **D. A. Castelow, D. W. Murray, G. L. Scott, and B. F. Buxton**. "Matching canny edgels to compute the principal components of optic flow". *Proceedings of the 4th Alvey Vision Conference, Manchester*, 1987.

[9] **J. L. Crowley, P. Stelmaszyk, and C. Discours**. "Measuring image flow by tracking edge-lines". *Second International Conference on Computer Vision*, 1988.

[10] **J. Canny**. "A computational approach to edge detection". *IEEE Trans. Pat. Anal. and Mach. Intel.*, 8:679–698, 1986.

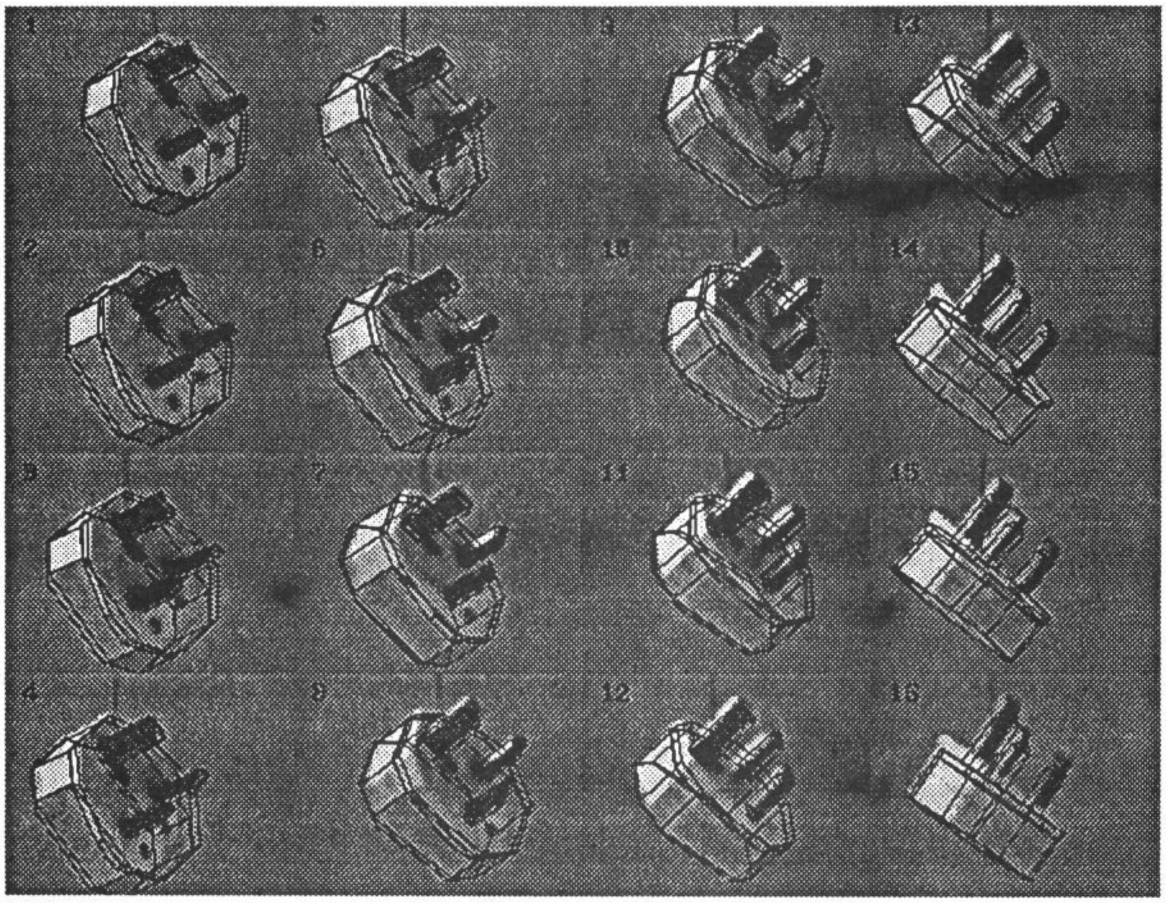

Figure 5: Sequence 2

[11] **P. Anandan**. "Computing dense displacement fields with confidence measures in scenes containing occlusion". *IUS Workshop, DARPA*, pages 236–246, 1985.