Tracking Curved Objects by Perspective Inversion

Alistair J Bray
School of Cognitive and Computing Sciences
University of Sussex

Abstract

A method is presented for tracking general curved objects through 3-space, given a sequence of grey-level images. The explicit recovery of 3D features is avoided and results demonstrate the method to be stable, accurate and robust. The object model has two parts — a tracking model and a grey-level model; the former specifies which features of the object are tracked, and the latter determines the appearance of these features. The method assumes initial position is known. Visible features are tracked using correlation between their rendered appearance and the next frame, to give a set of disparities. These disparities are used to invert the perspective transform and give the new position of the object.

It has recently been demonstrated that to track known, rigid planar-faced objects through grey-level image sequences it is unnecessary to extract 3D features from the images [1, 2, 3, 4, 5, 6]. Such work has shown that the approach of tracking 2D features and then inverting the perspective transform to recover 3D object position is not only highly robust to image noise and partial occlusion [3], but can also be implemented in real-time given current hardware [4, 5, 6]. Much of the above work concentrates upon tracking planar-faced objects, and uses only *minimal modelling*. In Bray's work — tracking a rotating plug through a sequence of "dirty" low-resolution images — robustness is achieved by using a small set of well-defined line features [2]; Stephens relies upon multiple viewpoints to achieve robustness [4]. Since Stephens and Harris both deal with real-time issues, their models consist only of a 3D-point set.

This paper demonstrates that more sophisticated modelling reaps benefits in terms of accuracy of results and the stability of the tracking routine. More precisely, a *grey-level modelling* approach allows correlation between the predicted appearance of the model and the actual appearance, giving disparity vectors that will act to reduce accumulating error in position. It is suggested that such grey-level models can be computed automatically (e.g. [7, 8]) and rendered in real time. The expense of this is partly offset by savings when inverting the perspective transform (since more consistent error vectors yield faster iterative solutions). The paper also demonstrates that the 2D approach provides a robust method of tracking rigid curved objects. All that is required for such tracking is that there exists a set of 3D features defining a model, that these features are precisely located, and that their visibility can be determined. However, further grey-level modelling is very useful for tracking these features through images. Finally, the methods are demonstrated on an image sequence that shows a real image rendered onto a rotating sphere. The sequence allows the estimated path in rotation/translation space to be compared with the true path.

1 Tracking Algorithm

The tracking algorithm used is described in detail in [3, Chapter 4]. The method is conceptually simple: project the model at the predicted position in the next frame to get a set of image features, track these image features into the next frame, and invert the perspective transform to determine that position (rotation and translation) that best maps the model onto the new set of image features.

Predicting Position

The general rule is that unless information is available concerning smooth motion in the past, the position in the last frame is the predicted position for the next frame. However, if motion over the past few frames has been smooth, then this can be used to guess a better position in the next frame. More specifically, the position in the first frame is taken as known. For the next six frames the estimated position p_t for f_t is p_{t-1}. After this the seven motion parameter curves (three for the translations in x, y and z and four for the rotation parameters in quaternion notation) are fitted with a quadratic curve. If the fit is good then the curve is extrapolated to give an estimated position in the next frame, otherwise the last position is used.

Tracking Features

To track features we use a primitive correlation algorithm that sums the absolute difference in grey-level intensity (See [3, Page 79]). Model features are projected at the expected position for the next frame, and the region around these features is rendered using the grey-level model. Correlation is then performed at these points only, *between the rendered patches* and the next image. The result is that the disparities represent the difference between the expected position in the next frame and the actual position.

Inverting the Perspective Transform

Lowe's algorithm for inverting the perspective transform is used [9] (described in [3, Pages 82–84]), although others could be adopted (e.g. [10, 4]). An error term is defined, based upon the disparity between a projected feature and its actual position. This total error ϵ, for each correspondence, can be expressed as a sum of the products of the error on each individual position parameter and the partial derivative of the error term with respect to this parameter i.e.

$$\frac{\partial \epsilon}{\partial d_x}.\Delta d_x + \frac{\partial \epsilon}{\partial d_y}.\Delta d_y + \frac{\partial \epsilon}{\partial d_z}.\Delta d_z + \frac{\partial \epsilon}{\partial r_x}.\Delta r_x + \frac{\partial \epsilon}{\partial r_y}.\Delta r_y + \frac{\partial \epsilon}{\partial r_z}.\Delta r_z = \epsilon$$

Considering error in the x and y dimensions independently, each correspondence yields two such equations in the six incremental variables Δd_x, Δd_y, Δd_z, Δr_x, Δr_y, Δr_z. Lowe's method is efficient, largely because it reformulates the projection equations to give simple, independent partial derivatives of the error term with respect to each of the six position parameters. If there are more than three correspondences between model and image the equations are over-constrained and a least-squares solution is adopted. In short, each iteration demands the least squares solution to a set of $2n$ linear equations in six variables, where there are n correspondences.

Motion Inertia

The *predicted position* is used for rendering model features and as a starting point for inverting the perspective transform. If it is accurate the feature

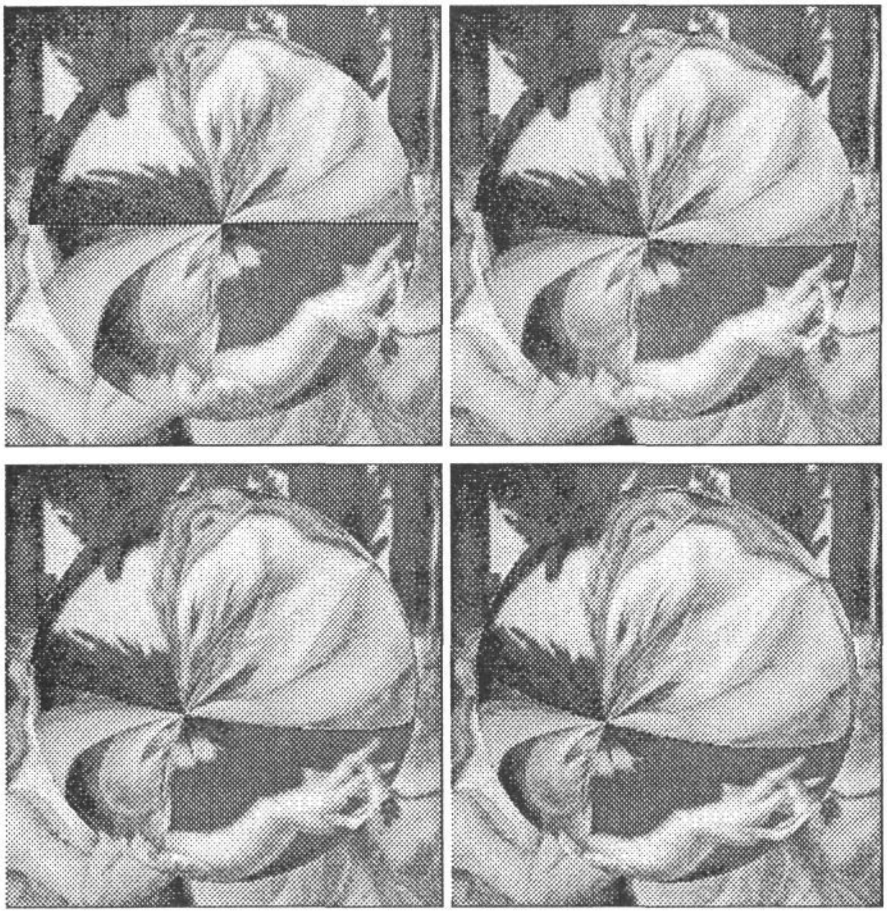

Figure 1: **Four Sample Images**

tracking should be accurate (due to the similarity between rendered and image feature) and fast (due to closeness in position). Inverting the transform should also be accurate and fast since the initial guess will be close to the final solution, and the disparity vectors will be consistent. However there is no *inertia* in the algorithm, and if the predicted position is inaccurate due to bad assumptions about smooth motion, the expected solution will still be found. If desired, it would be easy to incorporate motion inertia. When inverting the perspective transform both the least-squares error term and the number of iterations give measures of the quality of the final solution. These measures can be used to exclude bad solutions; in such cases, the estimated position computed by extrapolation of the motion curves can be used as the final position.

2 Images

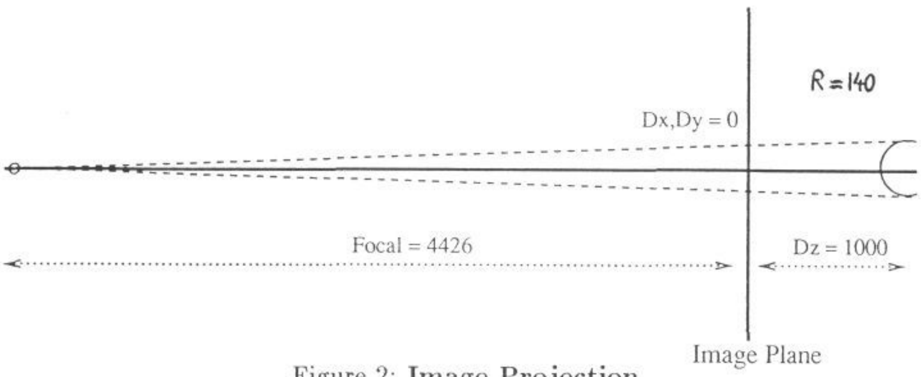

Figure 2: **Image Projection**

The first four images of the sequence are displayed in Figure 1; there are 72 such images (each 256x256x8). The sequence was generated by rendering a 256x256 grey level image onto a sphere of radius 140, and projecting the sphere into the image from a certain position $\{r_x, r_y, r_z, d_x, d_y, d_z\}$. The rendering algorithm parameterises the image as x, y (rectangular coordinates) and the sphere as θ, ϕ (angles of latitude and longtitude); it then maps patches of the image ($\int \delta_x \delta y$) onto patches of the sphere ($\int \delta_\theta \delta \phi$) using the same quantisation scales. The origin was set as the centre of the image plane and the focal point was (0,0,-4426). The translation remained constant at $\{d_x, d_y, d_z\} = \{0, 0, 1000\}$, but the rotation was defined as $\{r_x, r_y, r_z\} = \{5f^\circ, 5f^\circ, 5f^\circ\}$ where f is an integer and $0 \leq f < 72$. The rotations were composed in the order $r_z.r_y.r_x$. The rotation is therefore smooth and periodic, but not a constant rotation about any axis. To give an indication of the perspective effects involved in the sequence and to provide a yardstick against which the errors in Figures 5 to 11 can be judged, Figure 2 demonstrates a scaled version of this projection.

3 Models

- **The Tracking Model**
  The tracking model consists of a set of 3D points that are chosen such that they are easy to track in 2D. Line detection was run on the 256x256 image used to produce the image sequence. For each point in the image where two lines (thresholded on length) met, and where the angle between then was greater than 20°, the corresponding 3D point on the sphere was used as a model point. In the experiments described below there were 160 such points, of which about half were usually visible (and thus useful) in any frame. Therefore, about 80 points were used for inverting the perspective transform.

- **The Grey-Level Model**
  The grey-level model allows an image of the object to be rendered at any position. In our case, the grey-level model consisted of a rectangular image and the function that maps it onto the sphere. The information

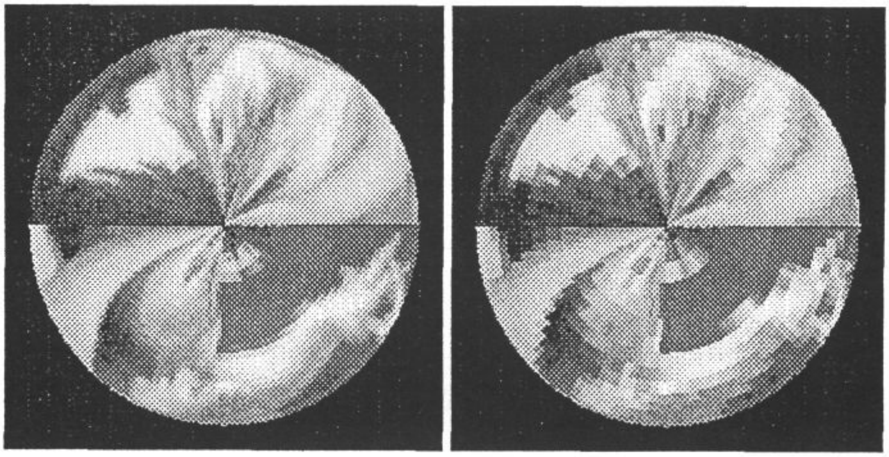
Figure 3: **Rendered images from two models**

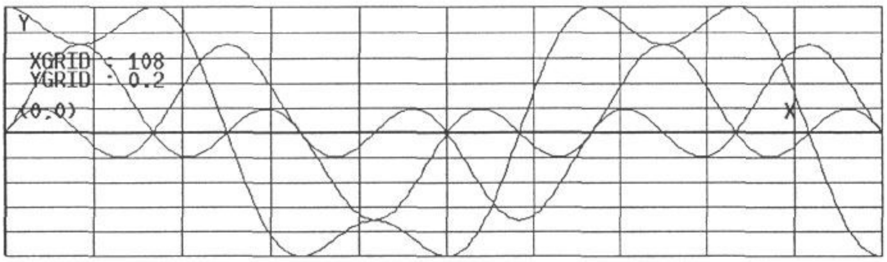

Figure 4: **Generating Rotation Parameters**

content of such a model is determined by the resolution of the image. In the experiments below we demonstrate results obtained using 2 resolutions, the lowest of which requires only 4.1K of memory.

4 Experimental Results

Two experiments were undertaken using grey-level models of different resolutions — 128 and 64. First frames generated from each model are displayed in Figure 3. Using 4-element quaternions to describe rotation, Figure 4 illustrates how the parameters should change over two complete revolutions[1]. We plot the tracked positions of the sphere and compare these with correct position. Results will be presented in terms of error in quaternion parameters and error in translation. The sphere was tracked for 3 complete revolutions i.e. 1080° about each axis. The mean and standard deviation of absolute error are presented in Figure 5 for both resolutions for each revolution, and for the three revolutions together.

[1]The q_x and q_z follow the same course for this path, and so there are only three visible graphs. Note that the cycle repeats after 720°.

	Cycle 1		Cycle 2		Cycle 3		All 3 Cycles	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD
RESOLUTION 128								
qr	0.003	0.003	0.004	0.004	0.003	0.004	0.003	0.004
qx	0.005	0.004	0.005	0.005	0.006	0.006	0.006	0.005
qy	0.006	0.006	0.006	0.005	0.006	0.005	0.006	0.006
qz	0.004	0.003	0.004	0.003	0.004	0.004	0.004	0.004
dx	1.228	1.015	1.126	1.016	1.315	1.333	1.223	1.134
dy	1.169	1.2	1.444	1.308	1.435	1.367	1.349	1.3
dz	22.542	15.872	23.736	14.006	23.392	14.302	23.223	14.758
RESOLUTION 64								
qr	0.007	0.008	0.008	0.011	0.006	0.007	0.007	0.009
qx	0.01	0.007	0.011	0.01	0.009	0.008	0.01	0.008
qy	0.011	0.01	0.01	0.009	0.008	0.006	0.01	0.009
qz	0.007	0.006	0.008	0.006	0.007	0.005	0.007	0.006
dx	2.168	1.748	2.028	1.9	1.83	1.49	2.009	1.726
dy	2.268	1.888	2.563	2.531	2.104	1.921	2.311	2.142
dz	37.877	32.971	47.076	40.037	36.684	35.981	40.546	36.74

Figure 5: **Error statistics for the two experiments**

- **Resolution 128**
  Results are shown in Figures 6, 7 and 10. Tracking is extremely accurate. From Figure 5 it is hard to detect significant deterioration over time.

- **Resolution 64**
  For the coarser resolution error is greater than before. All parameters are worse (See Figures 8, 9 and 11). However, it is again significant that their is no deterioration of tracking over time.

We conclude that the tracking is remarkably robust even when the rendering model is minimal, managing to follow the object along a complex path with large interframe displacements. It is also stable.

Sources of Error

There are two obvious ways to eliminate error by improving the correlation algorithm:

- **Sub-pixel disparities**
  The correlation algorithm is extremely unsophisticated, and returns only pixel disparities. A better algorithm (e.g. Nishihara's [11]) would examine the correlation surface to achieve sub-pixel accuracy. Improved accuracy in the disparities allows a better least-squares solution to be achieved faster when inverting the perspective transform.

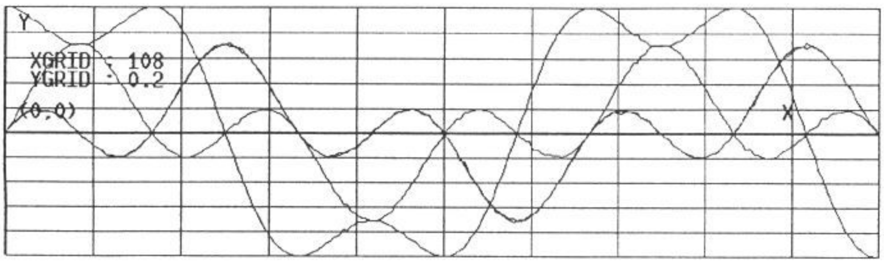

Figure 6: **Rotations: Resolution 128**

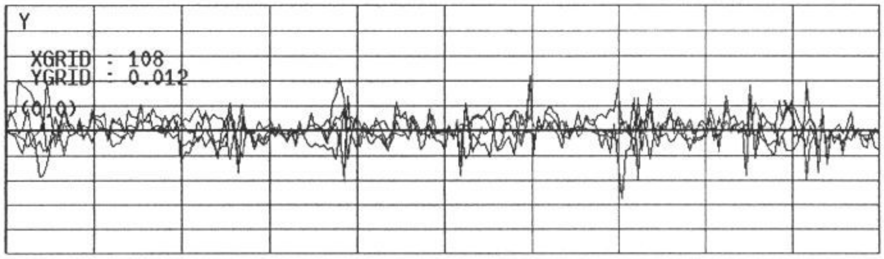

Figure 7: **Rotation Error: Resolution 128**

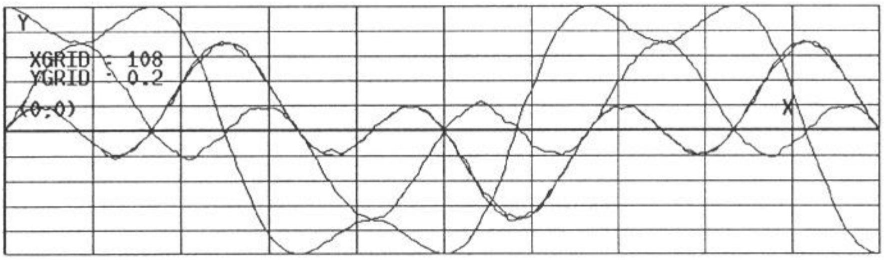

Figure 8: **Rotations: Resolution 64**

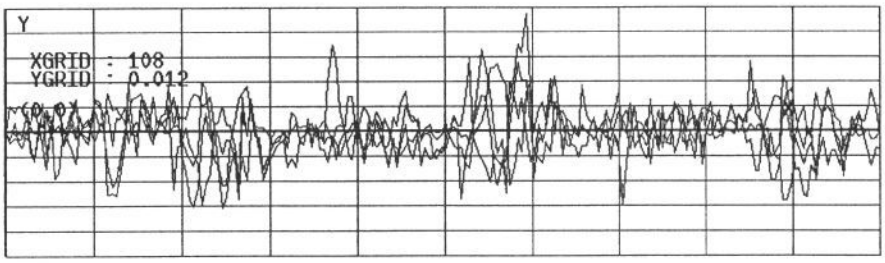

Figure 9: **Rotation Error: Resolution 64**

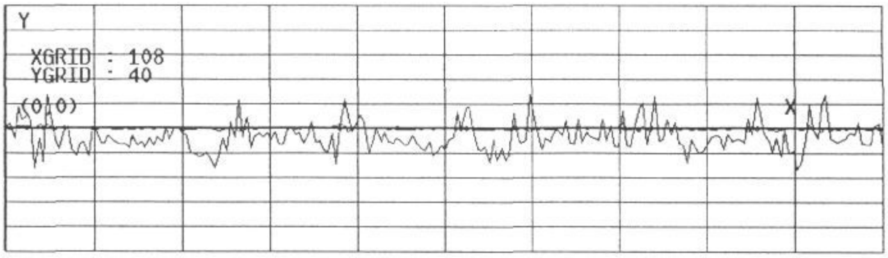

Figure 10: **Translation Error: Resolution 128**

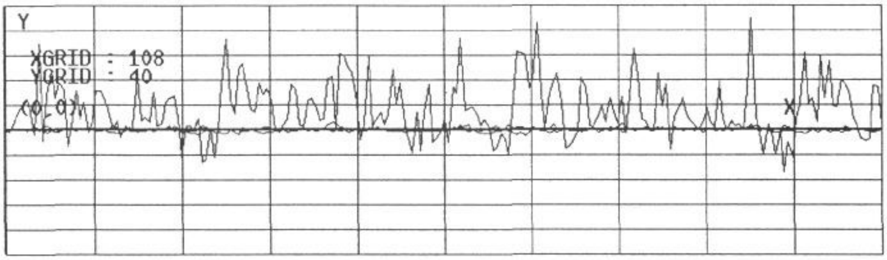

Figure 11: **Translation Error: Resolution 64**

- **Bad Matches**
  Examination of the correlation surface can also provide a measure of confidence in the match, determined by the "peakedness" of the correlation surface. Thresholding upon such a measure of confidence would allow dubious matches to be ignored and, as above, allow better solutions when inverting the transform.

5 Conclusions

These preliminary investigations are extremely promising. It has been demonstrated that grey-level modelling allows successful tracking of general curved objects (in many ways a sphere is the hardest case due to its symmetry) that are following non-trivial paths. The algorithm described here promises:

- **Stability**
  Since correlation is between predicted and actual appearance, the disparity vectors make the system stable i.e. tracking can improve after deteriorating (within the limits of the disparity window).

- **Accuracy & Robustness**
  The algorithm can provide quite accurate solutions using low resolution models, limited structural information, and inexact disparity information.

- **Speed**
  The algorithm is potentially fast since both tracking and grey-level models can be low-resolution, and rendering images, computing correlations and inverting matrices can all be performed by parallel algorithms.

The work described here was exploratory[2]. Extensions being considered are:

- **Grey-level modelling**
  Methods for generating these models automatically from real images are being studied by North [7, 8]. The use of non-uniform sampling (i.e. high density rendering around tracked features) is also being considered.

- **Real Images**
  The testing of these methods on sequences of real images is intended.

- **Faster Computation**
  The exploitation of good graphics algorithms is expected to improve the efficiency of the current system.

References

[1] **Bray A J**. Tracking models using convergence techniques. *CSRP 114, Sussex University, UK*, 1988.

[2] **Bray A J**. Tracking objects using image disparities. *Proceedings of the 5th Alvey Vision Conference, Reading*, 1989.

[3] **Bray A J**. *Recognising and Tracking Polyhedral Objects.* PhD thesis, School of Cognitive and Computing Sciences, University of Sussex, UK, 1990.

[4] **Stephens R S**. Real-time 3D object tracking. *Proceedings of the 5th Alvey Vision Conference, Reading*, pages 85–90, 1989.

[5] **Harris C and Stennettt C**. RAPID - a video rate object tracker. *Proceedings of the 1st British Machine Vision Conference BMVC*, pages 73–77, 1990.

[6] **Lowe D G**. Stabilised solution for 3D model parameters. *The First European Conference of Computer Vision*, 1990.

[7] **North P R J**. Reconstruction of visual appearance. *Proceedings of the 1st British Machine Vision Conference BMVC*, pages 205–210, 1990.

[8] **North P R J**. Optimal surface fusion (OSF). *Proceedings of the 2nd British Machine Vision Conference BMVC*, 1991.

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

[10] **Worrall A D, Baker K D, and Sullivan G D**. Model based perspective inversion. *Proceedings of the 4th Alvey Vision Conference, Manchester*, pages 13–18, 1988.

[11] **Nishihara H K**. Practical real-time imaging stereo matcher. *Optical Engineering*, 23(5):536–545, 1984.

[2] I would like to thank Peter North and Jim Stone for their many contributions.