Robust Object Tracking

Martin Armstrong   Andrew Zisserman
Department of Engineering Science
University of Oxford, Oxford OX1 3PJ, England
email:{*mna,az*}*@robots.ox.ac.uk*

Abstract

We describe an object tracker robust to a number of ambient conditions which often severely degrade performance, for example partial occlusion. The robustness is achieved by describing the object as a set of related geometric primitives (lines, conics, etc.), and using redundant measurements to facilitate the detection of outliers. This improves the overall tracking performance. Results are given for frame rate tracking on image sequences.

1 Introduction

Object tracking through image sequences is typically achieved by concentrating only on boundaries, such as apparent contours. This reduces the computational demands, and allows frame rate tracking on non-specialised hardware. A number of such methods have been demonstrated: Harris [4] and Lowe [7] track rigid objects with a known 3D model, snakes [5, 2] have been used to track the image contours of deformable object, while deformable templates [6] have been used for objects with known or constrained geometry.

Our objective here is to make tracking robust to a number of ambient conditions which typically degrade performance. These include partial occlusion, photometric changes (shadows, lighting), camera modelling errors, and incorrect edge matching.

We increase the robustness of the tracker by describing an object at a number of levels, and at each level we have a model/hypothesis for which redundant measurements are available. These redundant measurements allow the hypothesis to be verified or refuted, and also allow the identification and removal of outliers. Outliers in this context are grossly incorrect measurements or associations, whose errors have a very large adverse affect on the accuracy of subsequent calculations [11].

The object is described by a set of related **geometric primitives** (here after called primitives). Primitives have a known geometry, e.g. lines and conics. At a low level the primitives are associated with a set of high contrast edges, and are used to reject outlying edges measured in the image. At a high level, the primitives are associated with the object pose, and are used to reject outlying model-image associations.

We also consider camera modelling error by utilising a set of different camera models (e.g. calibrated perspective, weak perspective, affine), so that the most appropriate model is used.

The robust methods are applicable to each of the different methods of tracking given above, but are demonstrated here by extending the RAPID tracker of Harris [4].

2 Background

The RAPID tracker represents a 3D object as a set of control points which lie on high contrast edges. The pose (position and attitude) of the object is estimated and the object outline tracked at field rate (50Hz) on general purpose hardware. The cycle of the RAPID algorithm is given in fig. 1. The algorithm is split into two parts, the first dealing with making measurements in the image, and the second calculating the new pose of the object.

Measurement The control points are projected onto the image using the predicted pose of the object obtained from the Kalman filter. For each control point, a 1D search is then executed to find the strongest image gradient in the vicinity of the control point, which is assumed to be the new position of the edge. The search used is a 1D canny edge detection. The search is perpendicular to the projected model outline in the image (see fig. 2).

Pose Update When correcting the pose of the object, it is assumed that the pose differs only by a small translation and rotation from the predicted position in 3D. The projection of the object onto the image plane is linearised about the predicted pose. The small pose correction can then be calculated directly using the position of the control points and the distance to the corresponding image edge. The actual pose correction is calculated by minimising the distance from the control point to the actual image edge found (see fig. 1).

The pose of the object is tracked over time using a Kalman filter. A constant velocity model is assumed for the object, so that there are 12 state variables (6 for pose and 6 for pose velocity) and 6 measurement variables (pose only).

However, RAPID has a number of drawbacks.

- Predict the position of the object
- For each control point i, find the strongest image edge, perpendicular to the projected model outline
- Calculate pose correction $\mathbf{q}$ using
  $\min_{\mathbf{q}} \sum_{i=1}^{i=n} (l_i + \mathbf{q}.\mathbf{c}_i)^2$
- Update the position of the object using a Kalman filter and $\mathbf{q}$

Figure 1: RAPID algorithm: $\mathbf{q}$ is the small pose correction; l_i is the perpendicular distance to the edge; and, $\mathbf{c}_i$ is a function of the position of the control point both on the object and in the image, and the orientation of the projected model outline.

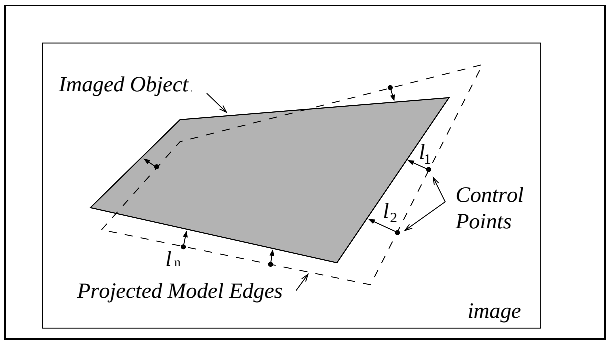

Figure 2: Measurements made by RAPID at the control points to correct object pose. At each control point a 1D search is carried out perpendicular to the projected outline to find the strongest image edge.

1. At no point are incorrect edges identified and eliminated. Incorrect edges arise from shadows, errors in pose causing alignment with erroneous edges, or texture on the object itself.
2. The stability of control points (how often/accurately they are found) is not utilised in any calculation.
3. The control points are treated individually throughout the algorithm, without taking into account that several control points are often placed on the same edge, and hence their measurements are correlated.
4. Only the calibrated perspective camera model is used.

We address these problems in the next two sections.

3 Robust Improvements

3.1 Robust Primitive/Object Detection

We define an object as a set of related primitives (high contrast edges which have a known geometry), which gives a model/hypothesis at two levels.

1. Finding the geometric primitives in the image.
2. Pose update based on the position of the primitives.

Using redundant measurements at each level allows verification of the models and the elimination of outlying measurements. The robust methods used for each level are described below.

Primitive Detection A mathematical model is known for each type of primitive (line, conic, etc). Sufficient control points are placed on each primitive to ensure redundant measurements (e.g. > 2 control points on a line, > 5 control points on a conic). A RANSAC [3] methodology is used to detect outliers among the edges detected by each control point. After the elimination of outliers, if the number of remaining edges falls below a threshold the primitive is not included in the pose update.

RANSAC The RANSAC algorithm is the opposite to conventional least squares techniques where as much data as possible is used to obtain a solution. Instead, as small a subset as is feasible is used to estimate the parameters, and the support for this solution measured. For example, in the computation of a line from several edges (see fig. 3), putative lines are estimated using random samples of 2 edges. The distance from the putative line to each edge is then calculated, and if it is below a threshold that edge is deemed to support the line. The process is repeated a set number of times, and the putative line with the most support is the one finally adopted. Outliers are edges that do not support the putative line chosen. After the elimination of outliers, the position of the line is then re-estimated using a least squares fit to the remaining (inlying) edges.

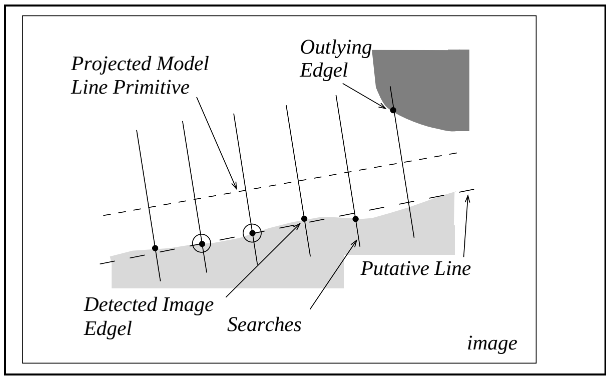

Figure 3: Detection of outlying edges for a line primitive. The outlying edge does not support the chosen putative line, which is fitted through the 2 circled edges.

Pose Update The pose update is robustly calculated using the remaining primitives, and outlying primitives are detected and eliminated. A typical outlying primitive is shown in fig. 4. Most of the primitives will have been found correctly, so rather than use a RANSAC methodology, case deletion is used to detect the outlying primitives. Each primitive is deleted in turn, and the pose correction is calculated from the remaining primitives to give a putative corrected pose. A projection error is measured for each putative corrected pose, and if the largest error is significant, the primitive deleted in the calculation of that putative pose is considered outlying and hence eliminated.

The projection error is the image distance between the projected and measured primitive. It is measured only for the primitive deleted in the pose computation (similar to the Cross-Validation method). If a primitive has been incorrectly identified in the image (i.e. an incorrect image-model association) the projection error will be high.

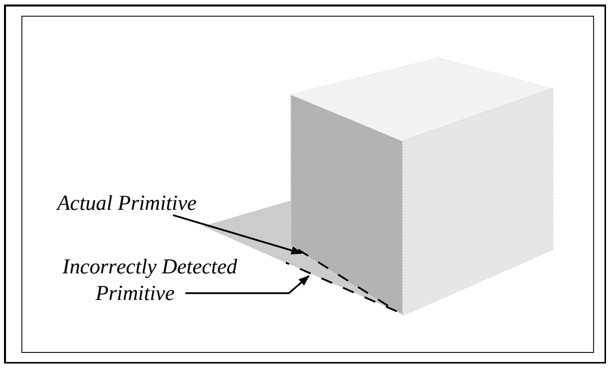

Figure 4: Detection of an incorrect primitive. The lighting condition results in a strong shadow near the predicted position of a line and causes the detection of an incorrect primitive.

Primitive Stability The stability of the primitives will vary over time, due to position of the object, lighting conditions, etc. Consequently, the confidence attached to a particular primitive should also vary. We compute this confidence from a decaying average of the frequency with which the primitive is correctly found. This confidence is then used to weight the affect of the primitive's control points in the subsequent pose correction calculation.

3.2 Camera Models

The original RAPID used a calibrated perspective camera. This camera model is a projection from object based coordinate $\mathbf{X}$ to image coordinate $\mathbf{x}$ given by $\mathbf{x} = \mathtt{C}\,[\mathtt{R}|\mathbf{t}]\,\mathbf{X}$, where $\mathtt{C}$ contains the camera intrinsic (calibration) parameters[1], and $\mathtt{R}$ (rotation matrix) and $\mathbf{t}$ (translation 3 vector) contain the pose of the object with 6 degrees of freedom (dof's).

The weak perspective camera is a useful approximation to the perspective camera [9], which only requires the aspect ratio of the camera to be known (rather than all 5 intrinsic parameters). The approximation requires a small field of view, and the object to have shallow relief compared with its distance from the camera. The average depth of the object is combined with the focal length to give the scale factor k.

The camera model is $\mathbf{x} = k\,[\xi \mathbf{r}_1 \mathbf{r}_2]^\top\,\mathbf{X} + k\mathbf{t}$ where ξ is the aspect ratio, $\mathbf{r}_{1,2}$ are the first 2 rows of $\mathtt{R}$, and $\mathbf{t}$ (2 vector) is the projection of the origin of object coordinate frame onto the image. $\{\mathbf{r}_{1,2}, \mathbf{t}, k\}$ are the pose of the object with 6 dof's.

The affine camera is the uncalibrated version of the weak perspective camera (i.e. aspect ratio is unknown). The camera model is $\mathbf{x} = \mathtt{M}\mathbf{X} + \mathbf{m}$, where $\{\mathtt{M}, \mathbf{m}\}$ are the pose of the object with 8 dof's. The affine camera is useful because an affine model can be generated automatically from an uncalibrated perspective camera when the relative motion is pure translation [8] (see below).

All three camera models have been implemented in the tracker. The general structure of the tracker is independent of the camera model used, only different pose parameters are tracked, and a different object to image projection used.

4 Implementation

The algorithm used for the robust tracking is given in fig. 6.

Model Acquisition A model is constructed from a set of primitives, which are the high contrast edges associated with surface creases, apparent contours, or surface markings. Currently the primitives have to have a known geometry (e.g. line, conic), to allow the placing of control points along their length, and the detection of outliers. Also defined in the model are the faces (opaque surfaces) which control which primitives are visible in the image, thus allowing hidden primitive removal. These are defined as an area enclosed by one or more primitives. Currently, the models are obtained using two images of the object from a calibrated (uncalibrated) camera to get a metric (affine) model [1]. Fig. 5 shows two such images, with the lines found in the image highlighted, and the actual wire frame model constructed.

Hidden Primitive Removal To detect which primitives are hidden (self-occluded) from a particular view, the primitives and faces are ordered according to distance from the camera. Then each primitive is checked to see if any, or all, of it is occluded by a face between itself and the camera. For a calibrated camera this can be done off-line for all possible view directions, and the visibility of each primitive stored in a table.

Motion Models The robust tracker and RAPID assume a general 3D motion model with 6 dof's. However, in many situations a more restricted motion occurs. We have implemented a number of restricted motion models: (1) ground plane motion, (2) pure rotation about the object centre of gravity, and (3) pure translation. In all cases the pose has fewer parameters than the 6 used in general 3D motion, which results in improved performance from the Kalman filter.

5 Results

These results were obtained from the tracker running at 50Hz on a SUN IPX workstation with an S2200 image board

[1]The 5 intrinsic parameters are focal length, aspect ratio, image skew, and position of the principle point.

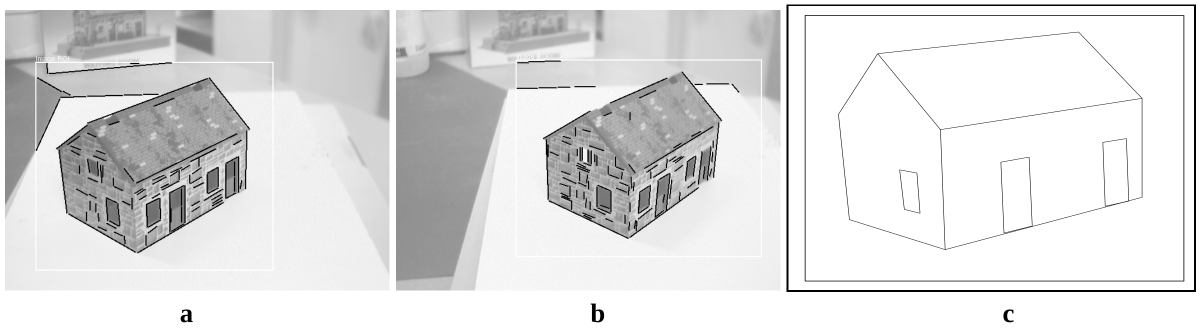

Figure 5: Obtaining a model of a object: (a),(b) the two images used, (c) the wire frame model. Note the internal (texture) primitives.

used purely for the image input/output. When tracking an object in a simple, uncluttered environment, the original RAPID generally works as well as our robust tracker. However, as soon as the object becomes partly occluded (see fig. 7), the original RAPID fails and starts to track a phantom object.

Fig. 8 shows the variation of pose parameters for both trackers, on a sequence where the object becomes partially occluded. RAPID fails, while the robust tracker continues to track correctly until the object moves out of view of the camera. Similar results are obtained for both the perspective and weak perspective camera models. However, the affine camera model is less stable, because the two extra degrees of freedom, which remove the rigid motion constraint, mean that quite often the primitives found do not fully constrain the pose correction (i.e. excessive skew is produced).

6 Open Questions

There are several interesting questions resulting from this work.

- Can the self calibration technique suggested by Quan [10] be used to transform from the affine camera to the weak perspective during normal tracking?
- How should models be updated/extended during normal tracking, to give more primitives to track?
- The tracker can detect when object “lock” has been lost. What are effective and efficient strategies for recovering the object in this situation?

Acknowledgements We are grateful for financial support from the EPSRC and ESPRIT BRA 6448 ‘VIVA’.

References

[1] Armstrong, M., Zisserman A., and Beardsley P. Euclidean reconstruction from uncalibrated images. In *Proc. British Machine Vision Conference*, pages 509–518, 1994.

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

[3] Fischler, M. A. and Bolles, R. C. Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 24:381–95, 1981.

[4] C. J. Harris. Tracking with rigid models. In A. Blake and A. Yuille, editors, *Active Vision*. MIT Press, Cambridge, MA, 1992.

[5] Kass, M., Witkin, A., and Terzopoulos, D. Snakes: Active contour models. In *Proc. International Conference on Computer Vision*, pages 259–268, 1987.

[6] Lipson, P., Yuille, A., Keefe, D.O., Cavanauch, J., Taffe, J., and Rosenthal, D. Deformable templates for feature extraction from medical images. In *Proc. European Conference on Computer Vision*, LNCS 427, pages 413–417. Springer-Verlag, 1990.

[7] Lowe, D. Integrated treatment of matching and measurement errors for robust model-based motion tracking. In *Proc. International Conference on Computer Vision*, pages 436–440, 1990.

[8] Moons, T., Van Gool, L., Van Diest, M., and Pauwels, E. Affine reconstruction from perspective image pairs. In *Applications of Invariance in Computer Vision*, LNCS 825. Springer-Verlag, 1994.

[9] Mundy, J. and Zisserman, A. *Geometric Invariance in Computer Vision*. MIT Press, 1992.

[10] Quan, L. *Self Calibration of an Affine Camera from Multiple Views*. Technical report, INRIA, 1994.

[11] Torr, P. *Outlier Detection and Motion Segmentation*. PhD thesis, University of Oxford, England, 1995.

```
• Predict the position of the Object
For each visible Primitive
    Calculate position of Control Points
    For each Control Point in Primitive
        find the strongest edge in image
    Eliminate outlying edges in Primitive using RANSAC
    If $number\ of\ edges > number\ of\ edges\ threshold$: then Primitive is found
• Correct pose using surviving Primitives/Control Points weighted according to stability
    using $\min_{\mathbf{q}} \sum_{i=1}^{i=n} w_i \left(l_i + \mathbf{q}.\mathbf{c}_i\right)^2$
For each surviving Primitive
    Delete Primitive, recalculate pose correction, and measure projection error
If largest projection error is significant: then eliminate Primitive
• Update Kalman Filter
```

Figure 6: Improved tracker algorithm, where w_i is the measure of primitive stability. Additions to RAPID are underlined.

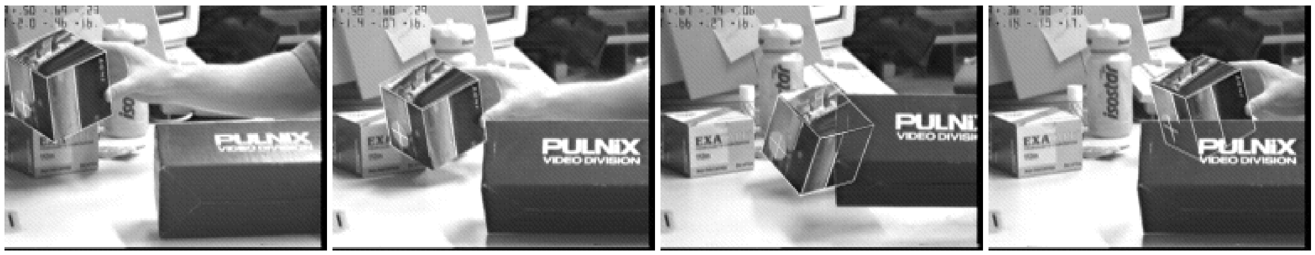

Figure 7: Images from a sequence when the object is partially occluded, but the robust tracker remains locked onto the object. Note, both internal (texture) and external (occluding) boundaries are tracked. The cross marks a tracked ellipse.

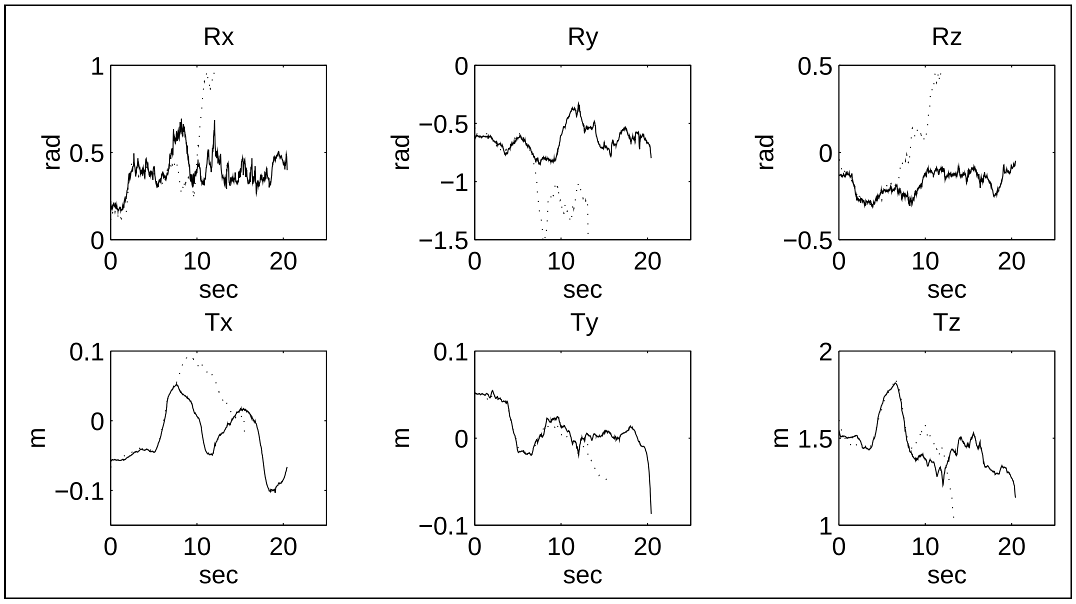

Figure 8: Object Pose Parameters for different trackers using the same image sequence: (1) Robust tracker (solid), (2) RAPID (dotted). The pose parameters are 3 for rotation of the object (given in the angle/axis form), and 3 for the translation from the camera to object coordinate frames. The object is partially occluded after 8 sec., causing RAPID to fail, while the robust trackers continues to track correctly until, after 20s, the object moves out of view of the camera.