Vision Guided Circumnavigating Autonomous Robots *

Nick Barnes and Zhi-Qiang Liu

Computer Vision and Machine Intelligence Lab, Department of Computer Science
The University of Melbourne, Parkville, Victoria, 3052, AUSTRALIA
nmb@cs.mu.OZ.AU

Abstract. In this paper, we propose a system for vision guided autonomous circumnavigation, allowing robots to navigate around objects of arbitrary pose. The system performs knowledge-based object recognition from an intensity image using a canonical viewer-centred model. A path planned from a geometric model then guides the robot in circumnavigating the object. This system can be used in many applications where robots have to recognize and manipulate objects of unknown pose and placement. Such applications occur in a variety of contexts such as factory automation, underwater and space exploration, and nuclear power station maintenance. We also define a canonical-view graph to model objects, which is a viewer-centred representation.

1 Introduction

In this paper we describe an autonomous robot system which is able to navigate around a given object. This circumnavigating system provides greater flexibility for robots that need to interact with objects, because only a model of the object is required, not of the environment, or position relative to the environment.

Our system identifies both the object and the view using a viewer-centred object representation. The inverse perspective transform of the object is then calculated, allowing the robot to geometrically generate a safe path around the object from the object model. The robot follows the path to the required relative location using odometry. It checks that each new view is expected given its belief about the object's position and its own relative position and motion. This increases the robustness of identification and prevents accumulation of the uncertainty inherent in navigation and odometry.

There are some difficulties in navigating around an object, for instance, background objects that are similar to the object concerned may confuse the robot resulting in moving to the wrong object, or wandering in an entirely wrong direction. Object recognition allows our system to discriminate similar objects. If the robot is required to navigate around one of several identical objects, our method is able to guide the robot by constantly updating its expectation about its position. Furthermore, deep, narrow concavities in the object can create difficulties for following the object's surface. A concavity that is marginally wider than the robot could cause a naive system to get stuck. However, a knowledge-based system should be able to reason about the size of the concavity relative to itself and determine how far it can safely move into such a concavity.

Our current system navigates around 3D objects with the robot moving on the ground plane. Camera elevation may be varied as long as the absolute height is known. As a result there are only three degrees of freedom (one rotational and two translational) for calculating the respective locations of robot and object. We assume that the robot is close enough to the object for recognition.

Our system is novel in the following ways:

* This work is supported in part by an Australian Research Council (ARC) large grant.

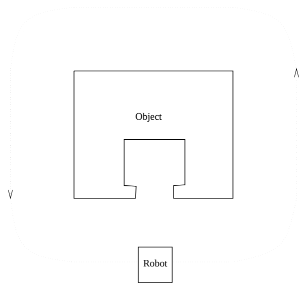


Fig. 1. Expected path for an object with a hole too narrow for safe robot access.

1. There is no need to model the environment, or the object's position in the environment. Robot navigation systems such as those proposed in [6] and [8] require a fix on object location within the world frame.
2. If the object is rotated, occluding the required surface, our system can navigate around it to find the required location. Arkin and MacKenzie [1] do not model the environment, but require that a specific object surface be visible.

2 System Architecture

Figure 2 shows the system architecture. There are four basic parts: pre-processing; view recognition; determination of the inverse perspective transform; and model-based geometric path-planning. Pre-processing takes a raw image and returns the required features, for instance, edge segments. View recognition matches the features to views in a *canonical-view* model, identifying the view in an image and correspondence between the image and model. Image-to-model correspondences are used to calculate the 3 degree of freedom inverse perspective transform using a direct algebraic solution. From a relative fix on object location the robot can plan a path around the object to attain its goal, then execute the movement of the planned path. Details of pre-processing and a derivation of the inverse perspective transform are available in [2].

2.1 Object Recognition Using Canonical-Views

Our object recognition system recognises both the object and the view in the image. We use a viewer-centred, model-based technique, similar to the aspect graph representation. Viewer-centred representations require only a 2D-2D match, significantly reducing the computational cost of evaluating candidate views.

Aspect graphs enumerate every possible *characteristic view* of an object, where a characteristic view is a continuous region of views, for which all visible edges and vertices, together with their connectivity relationships are the same [7]. Such changes in visible feature topology are called *visual events*.

Aspect graphs have problems of practicality [4]. A single surface may be redundantly represented in many separate views, which are differentiated only by visual events that may not be observable in practice. Also, there is no adequate indexing mechanism for these views. In the context of robot navigation, we may eliminate many views by applying the following observations:

1. Real cameras are not point but have finite size.

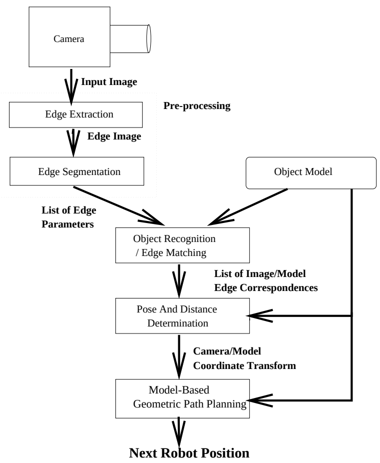


Fig. 2. System Architecture

2. Objects have features that contribute little to recognition.
3. Robots are not generally expected to recognise objects at very close range.
4. Camera images are discrete, thus for a particular viewing range, features below a certain size are not distinguishable.
5. Causality present in mobile robots plays an important role in recognition.

Definition 1 *A* canonical-view *is a set of neighbouring characteristic views: which are differentiated only by visual events involving features of size s for which $\frac{s}{l} \leq k$, where l is the size of the largest surface in the view, and k is constant; and which models only features that make a significant contribution to recognition.*

Note that k is set empirically to incorporate visual events such as the change from the front view to a side view of a wheel into a single view. The rule for the view allows recognition with either feature visible.

Definition 2 *A* canonical-view graph *has a node for each canonical-view of an object that is not a subset of the surfaces modeled for any other view; and has an edge between all view nodes that are neighbouring in the path of the robot.*

An aspect graph for the visible portion of the cube shown in Figure 3 would consist of 7 views. One possible canonical-view graph is derived by recognising that all views are subsets of view 7, and so only view 7 is represented. A matching scheme for such a representation would have to give a true match if any of the seven views occur. A second possible canonical-view graph would have nodes for views 1, 3, and 5. Here an image of view 7 would match all three canonical-views.

Figure 4 shows the form of the canonical-view graph for the model car pictured in the centre. The four views represent the adjacent car surfaces, and view ranges overlap at the corners of the car. Feature value ranges (discussed in the next section) were derived from segmented edge images of these views, with variance estimated using the perspective equations.

Causal Image Matching The system initially must check the image against every view in the model to find the best match, as object pose is unknown. However, once the first match is made and a move performed, the next view is *causally*

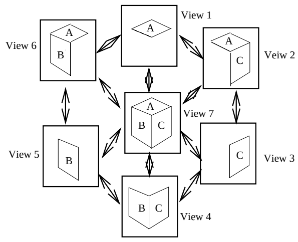


Fig. 3. The partial aspect graph for three neighbouring faces of a cube.

defined by the previous position and the movement. Each subgoal point, where the system will visually check its position, is linked to the canonical-view that the system expects to see. Thus, the system will generally only need to match one view for each move. Note that the combination of evidence from matching several *causally related* views greatly increases the certainty of identification.

Our canonical-view graph represents the causal relation between object views defined by the robot navigation problem described here, i.e. canonical-view graphs are indexed by order of appearance.

Features A canonical-view represents a range of actual views, therefore must allow a range for features. We use features similar to those discussed by Burns et. al. [3] for our canonical-view model: relative length of edges (Continuous range); orientation of edges (Continuous range); coincidental end-points of edges ([True, False]); and relative location of edges ([Above, Below], [Left-of, Right-of]).

We use edge orientation, η, relative to the image y-axis as there is only rotation about the Z axis, see Figure 5. Perspective variance of η is proportional to the angle, α, of the edge to the image plane, and the relative displacement, Δd, of the line end-points from the camera focal centre. Also, as α increases, foreshortening of the horizontal component of the edge, may η decrease.

The relative length of two edges varies when there is a greater change in the length of one edge between view angles. Edge length decreases proportionally to the distance from the camera, and proportionally to α.

End-points of two image edges are labeled coincidental if the ratio of the distance, d, between the nearest end-points and the length, l of the shorter of

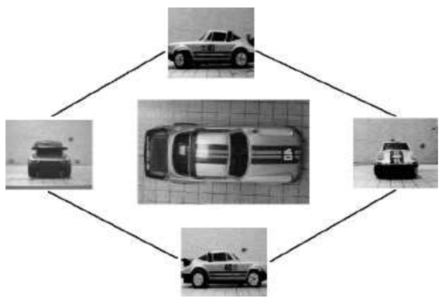
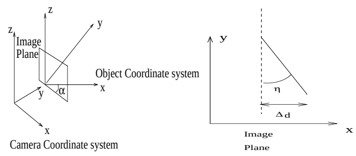


Fig. 4. The four canonical views of the model car used in our second experiment.



(a) (b)

Fig. 5. (a) The camera to object transform. (b) The image plane.

the edges is less than a constant c, for the range of the canonical-view: $\frac{d}{l} < c$.

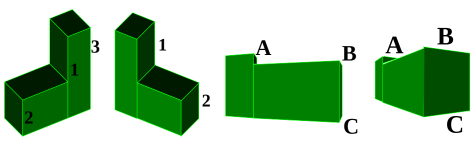
Relative edge location can be used as a feature due to the restricted rotation. A pair of edges is labeled if every point on one edge is above/below/left-of/right-of every point of the other, for every viewing angle. If a pair of edges has a left-of or right-of relation in one view it will be true for all views. This occurs except for cases where one edge is on a limb which is above the surface of the other edge, see Figure 6(a) and (b). Above/Below relations are affected by perspective. Parallel lines, coplanar in Z, will not cross, but, non-parallel edges or edges at varying depths may, (Figure 6(c) and (d)). The perspective effects vary with displacement from the image centre in x, and with α.

Feature ranges for canonical-views can be estimated by considering perspective projection of views which maximise variation. This can be empirically derived for each feature from images at extreme views, or estimated using the perspective equations for this restricted case, [2].

2.2 Model-Based Path Planning

To circumnavigate an object, the robot moves around the object in a single direction, following its surface, and remaining a safe distance away. The distance the robot must maintain from the object is problematic. Generally, it must be large enough for most of the object to fit into the view frame for recognition, but small enough for object points to be well spread in the image, allowing accurate pose and distance determination.

The robot generally moves to the next occluding boundary of the closest visible surface. Each move is performed as a view-based local navigation problem.



(a) (b) (c) (d)

Fig. 6. Relative spatial relations between edges (a) edge 1 is right-of edge 2, (b) edge 1 is left-of edge 3. Edge 1 is left-of edge 3 whenever both are visible, (c) Edge B and C are parallel and at the same depth, edge A is at a different depth, A is above B, (d) B is above A. B is always above C.

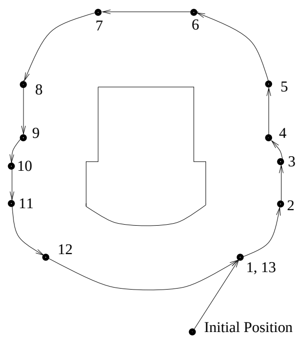


Fig. 7. Circumnavigation path derived for the terminal in the first experiment.

The robot determines the closest object point and the closest occluding corner point, in the direction of motion. If the corner is less than a threshold distance from the closest point to the robot, and the robot is approximately at the required distance from the object, then the robot is at the corner. The robot is guided by the surface normal and whether it is at a corner, as follows:

1. If not at a corner, move to be at the required distance from the object along the surface normal at the nearest corner in the direction of motion. See the path from Initial Position to (1), (Figure 7).
2. If at a corner, at the leading edge of a surface in the direction of motion, move at the required distance from the surface along the surface normal to the next occluding boundary. For example, the path from (2) to (3).
3. If at the corner, at the end of a surface in the direction of motion, move around the corner. This is done by subtending an arc with a radius of the required distance, and centred at the corner, finishing at the same corner on the next surface. For example, the path from (5) to (6).

Note that if a concavity is encountered, the path remains at least the required distance from all surfaces. From (3) to (4) (Figure 7) the robot does not move around to be perpendicular to the small surface behind the screen as that would bring it too close to the adjoining edge. Path generation is described in [2].

3 Experimental Results

We have conducted experiments using: a simulated image of a computer terminal, see Figure 8; and camera images of a model car, see Figure 9. The simulated images were generated using the POV ray tracing package.

3.1 Simulated Circumnavigation

The task is to circumnavigate a computer terminal, returning to the first surface encountered. Figure 8(a) shows a birds-eye view of the points where the system was expected to move, verses actual points determined by the system. Tabulated results for both our experiments are available in [2].

Note, Figure 8(a) shows no actual points for expected points between (3) and (4), and (7) and (8). If the system determines that it is at a corner, it progresses to the next corner. At point (3) the system determines that it is close enough

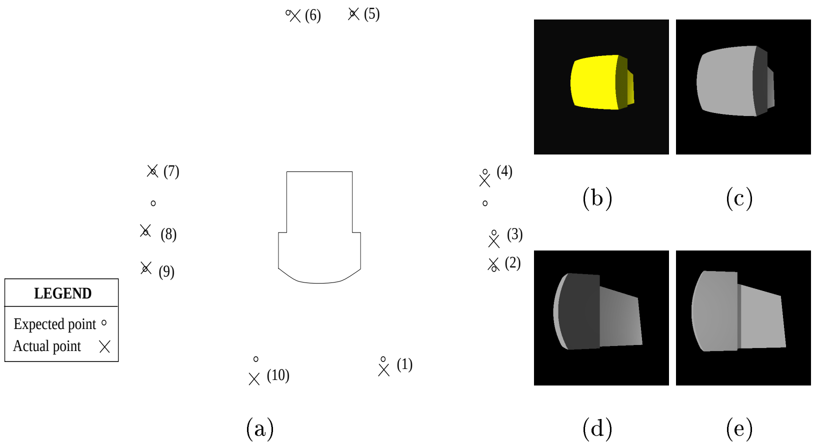


Fig. 8. (a)Expected and actual positions for the second experiment, (b) Initial view, (b) First view chosen, (c) Second view chosen, (d) Third view chosen.

to both the nearest corners, so it skips the second expected point entirely, and moves directly to point (4). Similarly at point (7).

This simulation demonstrates circumnavigation of an object. The system recognises the terminal, and generates a safe path around it, maintaining knowledge of its approximate relative pose and position.

3.2 Docking With a Model Car

Our second series of experiments uses a model car to demonstrate an application in an industrial automation setting. The robot is required to identify the car and navigate around it to the driver's side door (for Australian cars), where it is to move in close to the door's back edge for final docking. In doing this, it should determine the shortest path (clockwise or anti-clockwise) around the object, and pick the required car out of the two cars in the first image. The canonical-view model used for this experiment is shown in Figure 4.

Figures 9(b)-(e) show the camera views determined by the system. From 9(b) the system determines the number of points to be visited in either direction around the object: three moving clockwise; or seven anti-clockwise. Thus, clockwise is chosen, demonstrating high-level reasoning. As can be seen from Figure 9(a) the system was able to generate a safe path around the car, and arrive close to the required docking position.

4 Conclusion

We have implemented a vision system to guide an autonomous robot in circumnavigating objects of arbitrary pose. By applying 3D object/view recognition and pose determination the system was able to plan a path around the object, based on a geometric model, allowing the robot to navigate safely around a known object. The technique handles errors in position estimation and odometry by frequently visually checking its position relative to the object, and by restricting the distance moved based on any single estimate, relative to the robot's distance

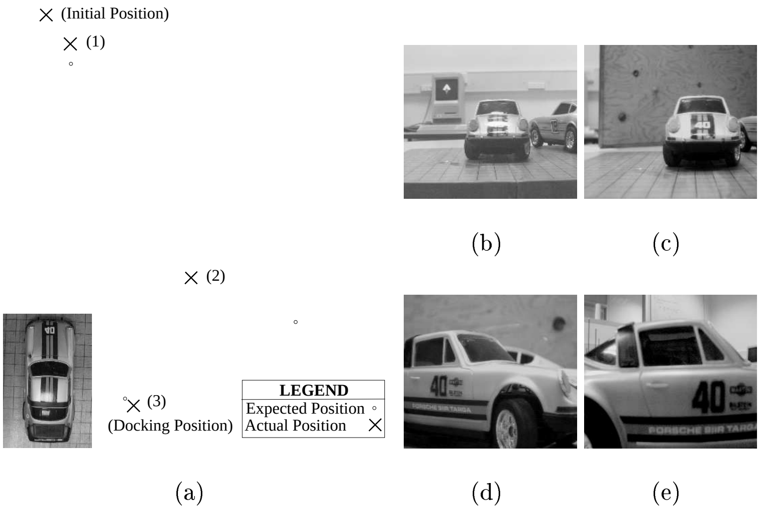


Fig. 9. (a) Expected and actual positions for the second experiment, (b) Initial view, (c) First view chosen, (d) Second view chosen, (e) Final docking position.

from the object. We also defined a canonical-view model for object recognition suited to mobile robots.

Our subsequent experiments include background objects blocking the robot's path. Further experiments will determine partial shape-from-shading to discriminate between objects with similar wire-frames, and allow partial modeling, so that the path can be planned based on the robot's determination of shape. Also, fixation techniques [5] need to be added to allow the system to fixate on a single object when there are similar or identical objects in the background.

References

1. R. C. Arkin and D. MacKenzie, Temporal coordination of perceptual algorithms for mobile robot navigation, *IEEE Trans on Robotics and Automation*, **10**(3) (1994).
2. N. M. Barnes and Z. Q. Liu, Model-based circumnavigating autonomous robots, Technical report, Dept. of Computer Science, Univ. of Melbourne. (1995).
3. J. B. Burns, R. S. Weiss, and E. M. Riseman, View variation of point-set and line-segment features, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **15**(1) (1993).
4. O. Faugeras, J. Mundy, N. Ahuja, C. Dyer, A. Pentland, R. Jain, and K. Ikeuchi, Why aspect graphs are not (yet) practical for computer vision, in *Workshop on Directions in Automated CAD-Based Vision*, 97–104 (1991).
5. D. Raviv and M. Herman, A unified approach to camera fixation and vision based road following, *IEEE Trans. on Systems, Man and Cybernetics*, **24**(8) (1994).
6. U. Rembold, The karlsruhe autonomous mobile assembly robot, in S. S. Iyengar and A. Elfes, editors, *Autonomous mobile robots*, IEEE Computer Society Press: California, **2**, 375–380 (1991).
7. N. A. Watts, Calculating the principle views of a polyhedron, in *9th International Conference on Pattern Recognition*, 316–322 (1988).
8. C. R. Weisbin, G. de Saussure, J. R. Einstein, F. G. Pin, and E. Heer, Autonomous mobile robot navigation and learning, *IEEE Computer*, **22**(6) (1989).