COLLISION-FREE PATH PLANNING FOR A VISION GUIDED CIRCUMNAVIGATING AUTONOMOUS ROBOT

Nick Barnes and Zhi-Qiang Liu

Department of Computer Science, The University of Melbourne, Parkville, Vic., 3052, Australia
nmb@cs.mu.OZ.AU

ABSTRACT

In this paper, we present a method for deriving and executing a collision-free path to allow a vision-guided robot to navigate safely around a known object. This method has been integrated into our autonomous navigation system, which allows a mobile robot to navigate around and, interact with objects of arbitrary pose and position. This method reduces the risk of collision with the object by restricting robot movements based on a lower bound estimate of its distance to the object. The distance estimate is derived from a single camera image. We also present a simple algebraic method for estimating the relative pose and position of the object. The integrated system can be used in applications where robots have to recognise and manipulate objects of unknown pose and placement. Such applications occur in contexts such as factory automation, underwater exploration, space exploration and nuclear power station maintenance.

1. INTRODUCTION

In this paper, we describe a system which enables an autonomous mobile robot to navigate around an object without collision by maintaining a minimum safe distance from the object. The robot frequently checks its position relative to the object, only moving distances less than the robot-to-object distance between checks. The robot-to-object distance is assessed using a vision-based approach which is described here. The method has been integrated with our existing vision-guided circumnavigating autonomous robot system [2].

The technique for range finding presented here is particularly applicable for a vision-guided system. In the case that the image has already been pre-processed for object recognition, and image-to-model correspondences are known, no additional image processing, or sensor fusion is required. Only algebraic expressions need to be evaluated to extract a reliable lower bound estimate of robot-to-object distance.

The combined system provides greater flexibility for robots interacting with known objects, because only a model of the object is required, not of the environment, or position relative to the environment, unlike systems such as that in [7]. Thus, no model modification is required if the environment changes. This system recognises a 3D object and view regardless of object pose and position using a canonical-view representation, assuming the robot is close enough for recognition. Unlike the system presented in [6] no particular object surface need be visible. The inverse perspective transform of the object is estimated using a direct algebraic solution. The robot plans a path around the object based on its current relative position, which the robot follows using odometry. We then calculate the minimum distance to the object. Provided the robot does not move further than this distance before it checks its position it will not collide with the object. After it has traveled the estimated safe distance, the system checks that each new view of the object is expected given the robot's motion, and its belief about its position relative to the object. This increases the robustness of identification, and prevents accumulation of the uncertainty inherent in navigation and odometry.

Our system guides ground-based robots, however, the camera may move vertically. As a result, there are four degrees of freedom (one rotational and three translational) for calculating the object's relative location.

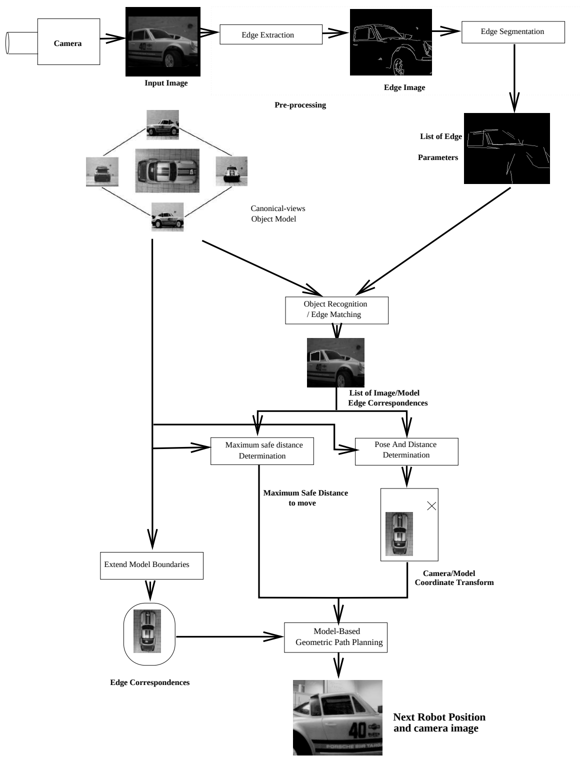


Figure 1: Architecture of Circumnavigating System

2. SYSTEM ARCHITECTURE

There are six basic parts in the system architecture (Figure 1): pre-processing; view recognition; determination of the inverse perspective transform; determination of minimum robot to object distance; model boundary extension; and model-based geometric path-planning. Pre-processing derives features, for instance, edge segments, from a raw image. The techniques used are described in [1]. View recognition matches these features to views, identifying the view and correspondence between the image and model. Image to model correspondences are used to calculate the inverse perspective transform, giving approximate pose and position of the object relative to the camera. The minimum robot-to-object distance is calculated to determine the distance the robot should move. The boundaries of the geometric object model are extended to generate a path around the object. From a relative fix on object location the robot can then reason about which direction it should travel around the object to attain its goal, and then execute movement along the planned path.

3. OBJECT RECOGNITION USING CANONICAL-VIEWS

Canonical-views are a viewer-centred, model-based object recognition technique, described in [1]. Essentially, canonical-views are a modified aspect graph [3] representation where: features that contribute little to object recognition are ignored; views that are redundant subsets of other views are eliminated; and neighbouring views are merged with rule-based variation if only a few small features differentiate them. The views are causally indexed by order of appearance to the robot as it travels around the object. The canonical-view representation used in our experiments is shown in Figure 1. Each of the four views shown represents the adjacent surface of the car. The canonical-views generated overlap at the corners. On first encountering an object, the robot must check the image against every view to find the best match. However, once a match is made and a move performed, the next view is causally determined by the previous position and movement.

4. OBJECT POSE AND DISTANCE

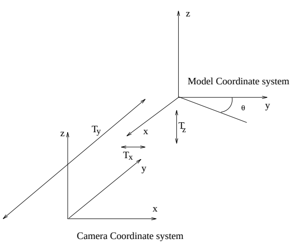


Figure 2: Model to camera coordinate transform.

Research such as that by Lowe [4] determines object pose from a single image with six degrees of freedom. Numerical solutions to pose determination are computationally intensive, and can have problems with stability, particularly with imprecise point correspondences. However, general robot navigation does not require precise pose, and, in our case, the object only has four degrees of freedom, so a direct solution is possible.

Previously, our technique assumed calibrated camera height relative to the object [1], however, sensitivity to slight tilt created difficulties for robot navigation.

By inspection of Figure 2 the perspective equations of image coordinates u and v for a pin hole camera, are:

$$u = \frac{f(T_x + Y_m \cos\theta - X_m \sin\theta)}{T_y - Y_m \sin\theta - X_m \cos\theta}, \quad (1)$$

$$v = \frac{f(Z_m + T_z)}{T_y - Y_m \sin\theta - X_m \cos\theta}, \quad (2)$$

where X_m, Y_m, Z_m are model coordinates, and T_x, T_y, T_z are translations along the x, y and z axes of the model coordinate system. Dividing (2) by (1) gives:

$$\frac{v}{u} = \frac{Z_m + T_z}{T_x + Y_m \cos\theta - X_m \sin\theta}. \quad (3)$$

If we consider edge segments instead of points:

$$\frac{\Delta v}{\Delta u} = \frac{\Delta Z_m}{\Delta Y_m \cos\theta - \Delta X_m \sin\theta}, \quad (4)$$

where $\Delta u = u_2 - u_1$, $\Delta v = v_2 - v_1$, $\Delta X_m = X_{m2} - X_{m1}$, $\Delta Y_m = Y_{m2} - Y_{m1}$, and $\Delta Z_m = Z_{m2} - Z_{m1}$, and (u_1, v_1) and (u_2, v_2) are two points in the image corresponding to (X_{m1}, Y_{m1}, Z_{m1}) and (X_{m2}, Y_{m2}, Z_{m2}) in the model.

However, $Y\cos\theta - X\sin\theta = \cos(\theta+\alpha)\sqrt{X^2+Y^2}$, where $\tan\alpha = \frac{Y}{X}$. Thus, we have:

$$\cos(\theta+\alpha) = \frac{\Delta Z_m \Delta u}{\Delta v \sqrt{X^2+Y^2}}. \quad (5)$$

Thus, the ratio of image to model vertical edge length is equal to the horizontal ratio, multiplied by $\cos(\theta+\alpha)$.

By considering Equation (2) for the vertical component of an edge segment we find:

$$\overline{T}_y = \frac{f \Delta Z_m}{\Delta v} + \overline{Y}_m \sin\theta + \overline{X}_m \cos\theta, \quad (6)$$

where $\overline{X}_m = (X_{m1} + X_{m2})/2$, and $\overline{Y}_m = (Y_{m1} + Y_{m2})/2$ are the mean model coordinates of the edge, and $\overline{T}_y = (T_{y1} + T_{y2})/2$ is the mean of the translations in Y to the model points (X_{m1}, Y_{m1}, Z_{m1}) and (X_{m2}, Y_{m2}, Z_{m2}).

Finally, we may deduce displacement in X, from the image centre, from Equation (1) by considering displacement of the vertical edge from (6), for $\overline{T}_x = (T_{x1} + T_{x2})/2$, the mean of the displacements to the model points:

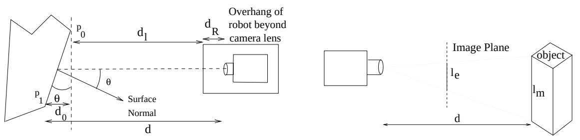
$$\overline{T}_x = -\frac{u}{f}(\overline{T}_y - \overline{Y}_m \sin\theta - \overline{X}_m \cos\theta) + \overline{Y}_m \cos\theta - \overline{X}_m \sin\theta, \quad (7)$$

Equation (5) gives only magnitude of θ. To determine sign, we find the closer of two edges from their relative ratio of image to model displacement in v, assuming perspective projection. For θ near zero, $\cos\theta$ and its sign become sensitive to edge length. If a one pixel change in edge length would reverse the sign estimate, we set θ to zero. For experiments we have conducted, this leads to approximately a 20° uncertainty in estimates of θ. This is acceptable for robot navigation tasks. However, the sign could be more robustly determined by analysing angle change over an image series. In practice, we use Equation (5) by taking averages of two displacements in u and v using four correspondence points. To estimate T_y, we assume calibrated focal length because it is interdependant with distance. Otherwise, the distance estimate will be sensitive to small errors in image to model point correspondences.

4.1. Collision Free Motion - Using a Lower Bound for Safe Distance from the Object

Our technique for estimating a safe distance to move is based on the vertical displacement of an edge, and its horizontal displacement from robot extremities. Vertical displacement is relatively invariant to robot motion around the object. Also, edge length errors need only be bound in one dimension. Note that estimation of distance discussed in the previous section does not provide a lower bound.

The derivation assumes: endpoints of segmented edges correspond to endpoints of actual edges; there are edges visible with significant vertical displacement between endpoints; point correspondences found during object / view recognition are based on correct edges; and a pinhole camera model.



(a) (b)

Figure 3: (a) Three components of camera to object distance (b) Distance to an edge from perspective equations

A lower bound, d_l, for distance from the object is defined (Figure 3 (a)):

$$d_l = d - d_R - d_0,$$

where d is the lower bound on distance to p_1, a point on the object, d_R is the upper bound on distance from the camera lens to the point on the robot closest to the object, and d_0 is the upper bound on distance from p_1 to p_0, the point on the object closest to the robot. d_R can be set to the largest possible overhang of the robot beyond the camera.

From the perspective equations, the distance d to an edge of known vertical displacement l_m, observed vertical displacement l_e, and camera focal length f is:

$$d = f l_m / l_e.$$

If the camera tilts by ϕ, d is foreshortened by $cos(\phi)$.

Let d_{p01} be the absolute distance from p_0 to p_1 in the model and θ be the angle between the image plane and the line joining p_0 and p_1, which is equal to the angle between the normal to this line and the viewing angle (Figure 3 (a)). We need to determine how much closer p_0 could be in the worst case, i.e. the additional horizontal displacement, d_0, beyond p_1 in the direction of the robot:

$$d_0 = d_{p01} * sin(\theta).$$

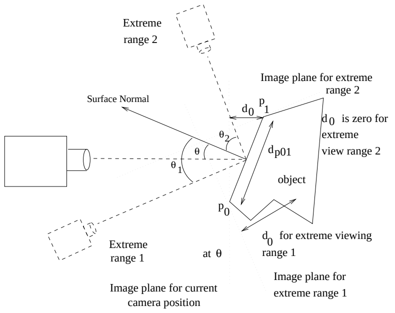
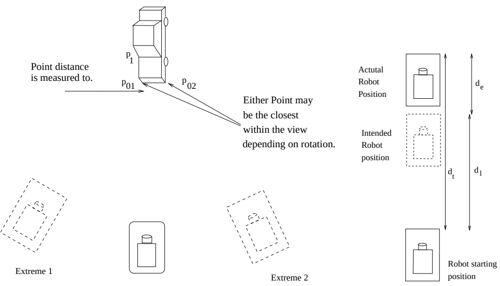


Figure 4: Displacement d_0 from p_1 to p_0 in the direction of the camera varies with respect to the viewing angle, between extremes of the viewing range.

However, viewing angle, and so d_0, vary within a canonical view, (Figure 4). Let the extreme views of the range form angles θ_1 and θ_2 with the normal to the line joining p_0 and p_1. The worst case (largest) d_0, for θ, where $\theta_1 \leq \theta \leq \theta_2$ occurs at 90°, if $\theta_1 \leq 90^\circ \leq \theta_2$, otherwise, the maximum is either θ_1 or θ_2. For these cases d_0 will be d_{p12}, $d_{p12} \cos \theta_1$ or $d_{p12} \cos \theta_2$, respectively. If there are several possibly closest points on the object, as in Figure 5 (a), the lower bound occurs at the point which maximises d_0.



(a) (b)

Figure 5: (a) Within a view several object points may be closer to the robot depending on the camera position within the view. From Extreme 1, point p_{01} is closest, from 2, point p_{02} is closest. (b) Distance moved: actual and intended.

4.1.1. Implementation

If the robot intends to move a distance d_l, but actually moves d_t, the error, d_e, can be defined $d_e = d_t - d_l$. d_e is a function of d_l, and variables particular to the robot and environment. If we calculate the maximum d_t then we can restrict d_l such that: $d_l \leq d_t - d_e$.

Edge segmentation errors can result in overestimation of distance to an object when an edge appears shorter than the actual, (Figure 6). However, if we make use of contextual knowledge from the model about object structure around the edges, e.g. edges that join, we can restrict the error in determining edge length. If both end-points of the segmented edge, e_1, join other segmented edges then our estimation of l_e should be correct. However, our matching process allows for imperfect segmentation, so structurally coincidental edge end-points may appear slightly separated. Let the observed separation between the endpoints be s_a.

If the angle, ϕ, between two edges with a common end-point is greater than $90°$ then the additional length l_a cannot be greater than s_a, see Figure 7(a) and (b). Thus, an edge with both endpoints coincident with other edges has maximum length d_e: $d_e \leq l_e + s_{a1} + s_{a2}$, where s_{a1} and s_{a2} are observed separations between the edge end-points and the joining edges. Thus, the lower bound on distance is:

$$d_l \leq fl_m/(d + s_{a1} + s_{a2}).$$

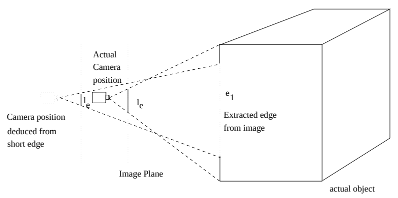


Figure 6: The extracted edge is shorter than the actual, so we may overestimate camera to object distance.

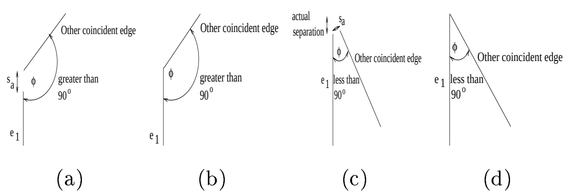


Figure 7: Segmentation errors for an edge pair with coincident endpoints: (a) segmented edges where $\phi > 90°$, (b) actual object edges, (c) segmented edges where $\phi < 90°$, (d) actual object edges.

For $\phi < 90°$, the additional length, l_{add}, may be greater than s_a, e.g. 7(c) and (d). Let α be the angle between the projection of the coincident edge, and the line joining their endpoints, see Figure 8. Using the Sine Rule we find:

$$l_{add} = (s_a sin(\alpha))/sin(\phi).$$

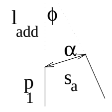


Figure 8: Applying the Sine Rule to edges with a coincident end-point, where ϕ is less than $90°$.

Note in Figure 5(a), the car windscreen has horizontal displacement as well as vertical. Our derivation is based on a point position of the edge. We estimate the minimum distance to the edge by taking the point which maximises d_0. Also, we must allow tolerance for image digitisation which can shorten observed edges.

Note, if structurally separate edges appear collinear due to viewing angle, l_e may be erroneously large, see Figure 9. However, this results in an underestimate of d, and so will not make robot moves unsafe.

4.1.2. Heuristics for line determination

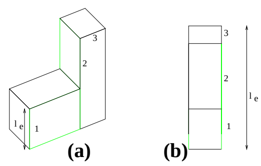


Figure 9: Edge 1 is clear in (a), however in (b) it is collinear with 2 and 3, and so appears to be longer.

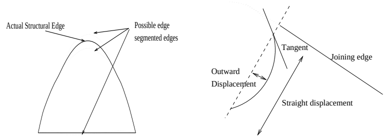
Although we require a lower bound estimate for d_l, if the estimate is unnecessarily small it will impede robot progress. We infer some heuristics to choose the best edge to base calculations on. We should choose edges: with structure around them (required to bound length); which result in the greatest estimate of total distance; and with large vertical displacement in the image to minimise digitisation error.

4.1.3. Maximum Vertical Span for Estimating Distance

The method above assumes that vertical edges, which are largely straight, are visible. If no vertical edges are visible, the object must have a curved surface with vertical span, see Figure 10 (a), (ignoring 2D objects). Ideally, the maximum vertical span is the distance between an extreme point at the top of the arc and the bottom edge. However, edge segmentation errors may result in span appearing smaller than it actually is. As no other edges join the top point, we do not have an effective bound on its location, and so must allow a tolerance for segmentation errors.

We define bendedness as the ratio of outward displacement over straight displacement, edges with low bendedness are prefered as they provide a more stable result. Figure 10 (b) shows a degenerate case where representing a curved line as a straight line results in underestimating vertical displacement. The error will be small if we calculate l_{add} based on the tangent at the end-point of the curve. Note, that if ϕ is very small, small bendedness may lead to a large error, however this will also be the case for a straight edge.

In practice, complex objects are likely to have structured edges with large values for ϕ. These should always be used to determine distance when available.



(a) (b)

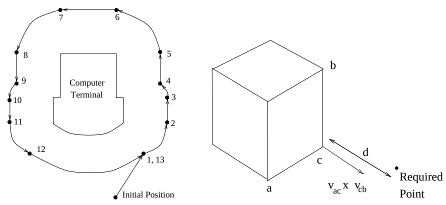
Figure 10: (a) No edges have structurally bounded vertical displacement. (b) Edges of high bendedness should be avoided, or l_{add} calculated from the tangent to the endpoint.

4.2. Model-Based Path Planning

Circumnavigation is the process of moving around an object, in a single direction, and returning to the first surface encountered, without colliding, or moving away from the object unnecessarily. To circumnavigate a robot should follow the object surface normal, executing a path that is at a minimum safe distance from the object. Circumnavigation can be performed as a series of view-based local navigation problems. At each point the robot's task is to move along the nearest surface, in the direction of circumnavigation.

The distance that the robot should remain from the object is task dependant. To dock a robot must move in close to a particular point after navigating around the object. We restrict the distance the robot can move based on its distance from the object. Therefore, the robot must be far enough from the object for each move to be a reasonable size. The distance also needs to be large enough to fit most of the object into the image frame for recognition, but small enough for object points to be well spread in the image, allowing accurate pose and distance determination.

We plan a path around the object *a priori* by extending object boundaries in two dimensions, similar to the *configuration space* approach [5]. Figure 11 shows a path derived in previous work [2]. This is a continuous path of points where it would be suitable for the robot to assess its relative position. To circumnavigate, the robot calculates the distance to move: the lower bound on robot to object distance minus a safety margin. It finds the intersection of a circle of radius of the moving distance, centred at its current position,, with the derived path, in the direction of robot motion. As the robot never moves further than the robot-to-object distance, it may move safely in a straight line to the chosen destination. The robot will not collide with occluded objects as these must be further away than the object. The system presented in this paper assumes no obstacles block the robot's path. If obstacles were present, the robot could still generate the ideal path *a priori*, and handle obstacles reactively when encountered. Note, in Figure 11 at point (4) and (9), that if a concavity in the object is encountered, the generated path remains at least the required distance from the surface at all times.



(a) (b)

Figure 11: (a) A derived circumnavigation path. (b) Surface normal at c from the vector cross product.

4.2.1. Deriving Expanded Boundaries for Surfaces

Objects are modeled by piecewise planar and quadric surfaces. A planar surface is represented as a list of connected boundary points. We extend, along the surface normal, the two points a and c, which form the widest projected edge in the plane of robot motion, and join them to form the expanded boundary. To extend c, we take the cross-product of vectors from two other points on the surface a, b to c:

$$\vec{v}_{ac} \times \vec{v}_{cb},$$

project it into the (x,y) plane, and take the unit vector, with components $\hat{x}$, and $\hat{y}$. For the point $c, (x_c, y_c)$, the projected point at the required distance d is:

$$((x_c + \hat{x} \times d), (y_c + \hat{y} \times d)),$$

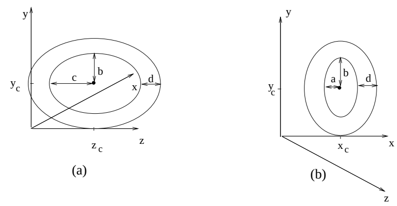


Figure 12: The expanded ellipsoid.

We represent elliptical surfaces as a centre point (x_c, y_c, z_c), and radii (a, b, c) in (x, y, z). Only convex non-rotated ellipsoids are handled presently. See [2] for an experiment using an object with elliptical surfaces.

$$\frac{(x + x_c)^2}{a^2} + \frac{(y + y_c)^2}{b^2} + \frac{(z + z_c)^2}{c^2} = 1$$

An approximation to expanded ellipsoid is formed by expanding each radius by d, see Figure 12. The widest projection of the ellipsoid into the (x, y) plane is where the absolute value of Z is a minimum, z_{min}:

$$\frac{(x+x_c)^2}{(a+d)^2} + \frac{(y+y_c)^2}{(b+d)^2} + \frac{(z_{min}+z_c)^2}{(c+d)^2} = 1$$

We form arcs of radius d around corners between adjacent surfaces to join the projected corners of the surfaces, see Figure 11 (a). Finally, any lines which closer to the object than other lines or set of lines are discarded. In Figure 11 (a), note that the two short, collinear boundaries, behind the curved front surface (screen) have been subsumed between points 5 and 6. Any intersecting lines are cut at the intersection, see where the arc 3-4 joins edge 4-5.

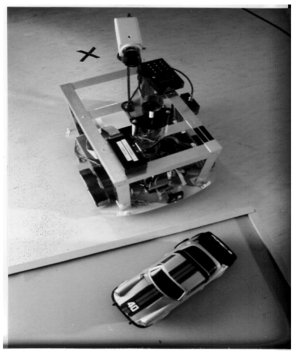


Figure 13: Our robot navigating around a model car.

5. EXPERIMENTAL RESULTS

We conducted an experiment where our robot navigates around a model car and docks at the rear right hand corner. The experiments used images from a Minitron Javelin camera, against the background of the Computer Vision and Machine Intelligence Lab (CVMIL) at the University of Melbourne. This experiment demonstrates how the system could be used in an industrial automation setting. The system must identify the car and move around it in a clockwise direction, and then move in close to the rear right hand corner for docking. The system is able to recognise the car and its relative pose and position, and to generate a safe path around it, always maintaining knowledge of its approximate relative position. Figure 14 shows the results of the experiment.

6. CONCLUSION

We implemented a method that maintains a safe distance for an autonomous mobile robot navigating around a known object. This paper presented a novel method for determining object pose, and a technique for estimating a lower bound on the distance to the object, coupled with a path-planning method to enable collision-free circumnavigation of an object.

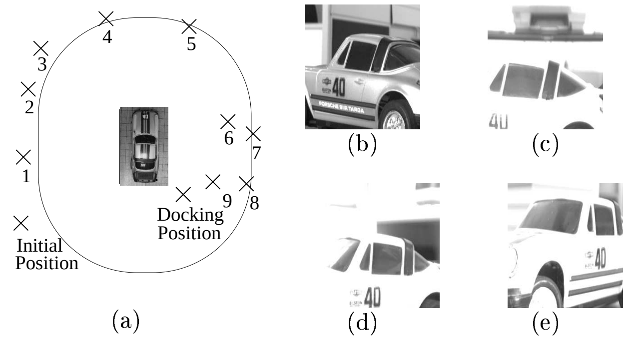


Figure 14: (a) Expected positions verus ideal path (the solid line) around the object. The actual docking position is the rear right hand corner of the car. (b) Image from initial view, (c) - (e) first three views chosen by the robot.

These methods form part of an ongoing project to develop a vision-guided system which can reliably interact with known objects of unknown pose and position.

7. REFERENCES

[1] N M Barnes and Z Q Liu. Model-based circumnavigating autonomous robots. Technical Report 12, Dept. of Computer Science, Univ. of Melbourne., 1995.

[2] N M Barnes and Z Q Liu. Vision guided circumnavigating autonomous robots. In *ICSC '95 - Proc. Third International Computer Science Conference: Image Analysis Applications and Computer Graphics, Hong Kong*, pages 33–42, 1995.

[3] J J Koenderink and A J van Doorn. The internal representation of solid shape with respect to vision. *Biological Cybernetics*, Volume 32, pages 211–216, 1979.

[4] D G Lowe. Three-dimensional object recognition from single two-dimensional images. *Artificial Intelligence*, pages 355–395, 1987.

[5] T Lozano-Perez. Spatial planning: A configuration space approach. *IEEE Trans. on Computers*, Volume C-32, Number 2, pages 108–120, Feb. 1983.

[6] D L Vaughn and R C Arkin. Workstation recognition using a constrained edge-based hough transform for mobile robot navigation. In *Proceedings SPIE:Sensor Fusion III:3D Perception and Recognition*, pages 503–514, 1990.

[7] C R Weisbin, G de Saussure, J R Einstein, F G Pin and E Heer. Autonomous mobile robot navigation and learning. *IEEE Computer*, Volume 22, Number 6, pages 29–35, June 1989.