Object Location and Recognition using Whisker Sensors

R. Andrew Russell and Jaury Adi Wijaya
Intelligent Robotics research Centre
Monash University
Clayton, VIC 3800, Australia
{andy.russell,jaury.wijaya}@eng.monash.edu.au

Abstract

The project described in this paper has the aim of demonstrating that whisker sensors can be used as a rapid and effective form of robot sensing. Previously a single whisker sensor has been demonstrated mounted on a mobile robot for gathering information about the surface profile of objects close to the robot. This work has now been extended to an array of eight whiskers mounted on a robot that also incorporates a simple gripper. The passive whisker sensors are scanned over external objects by the motion of the robot. By monitoring the resulting deflection of the whiskers sequences of surface points can be recorded. Object recognition algorithms have been developed that allow the robot to recognise, grasp and retrieve a range of objects using the whisker data. In this paper the upgraded robot, WhiskerBOT, is described together with the object recognition and localisation algorithms. Results of practical experiments are also presented.

1 Introduction

In nature, whiskers are a widely utilised sensory structure across many species of animals and even some plants. Insects and crustaceans employ whiskers and antennae to position their exteroceptors outside their tough exoskeletons so that they can sense the surrounding environment. Most mammals have body hairs and whiskers that can act as close proximity sensors. Rodents such as rats and mice together with nocturnal predators including cats have highly developed tactile whisker sensory systems. A cat captures and manipulates its prey using its mouth. Objects held in the mouth are too close for the cat's eyes to accommodate and are also partially obscured from view. Whiskers allow the cat to determine the position and movements of prey animals close under the nose or held between the jaws. The upper lip of the cat is well supplied with vibrissae. Each whisker has about 200 sensory nerve endings, which respond to displacement and velocity stimuli. Muscles position the whiskers to sense objects close to the mouth of the cat or fold them back to keep them out of the way while eating. When springing onto its prey, or when carrying something, the whiskers are angled far forward to envelop the object. Some idea of the speed and effectiveness of a cat's whiskers can be gathered from the ability of a blindfolded cat to locate a mouse. Within one tenth of a second of a mouse touching the cat's whiskers the cat grasps the mouse with a fast and precise bite to the nape of the neck [Layhousen, 1979]. This example demonstrates the speed and discrimination exhibited by biological whiskers.

There are many examples of the application of whisker sensors in mobile robotics. Long antennae-like whisker sensors were mounted on the SRI mobile robot Shakey [McKerrow, 1990] and on Rodney Brook's six-legged robot insects [Brooks, 1989] to provide warning of obstacles. Even some of the robots designed to participate in the Micromouse competition employed whisker sensors [Dibley, 1984]. Legged robots need to monitor the separation between each foot and the ground to allow deceleration of the foot before contact. It is also desirable to detect obstacles as the foot is swung forwards so that it can step over the obstacle. The four-legged robot Titan III has feet fringed with curved whiskers, which detect both ground proximity and obstacles [Hirose, et al., 1985]. These whiskers are made from shape-memory alloy because the high elasticity of this material allows it to tolerate a relatively large amount of bending without suffering permanent deformation. Similarly shaped whiskers have been considered for the legs of the Ohio State University active suspension vehicle [Schiebel, et al., 1986]. More recently Jung and Zelinski [Jung and Zelinski, 1996] report using passively articulated whiskers mounted on the side of their Yamabico robot to assist close wall-following behaviour. None of these applications show the speed and discrimination that animals demonstrate with their biological whiskers. The aim of this project is to take a step towards the development of robotic whisker sensors that can rival the speed and discrimination of biological whiskers.

2 The Whisker Sensor

As described in Wijaya and Russell [Wijaya and Russell, 2002] the whisker sensors used in this project consist of a straight rigid wire 20cm long and 1.6mm diameter mounted on a low friction servo-potentiometer. Two springs return the whisker to its initial position when it is not in contact with an object. A 4mm diameter plastic ball is attached to the tip of each whisker to improve its ability to slide over a range of surfaces. Whiskers rotate

in the *x,y* plane and gather 3D surface information with a constant z component (the z-coordinate of the whisker).

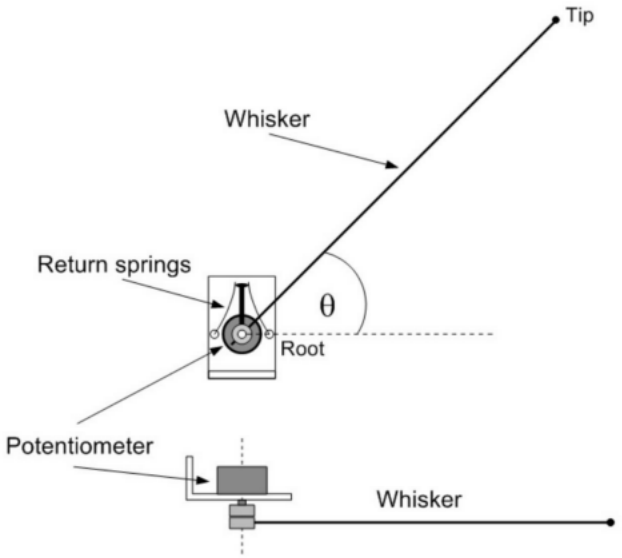


Figure 1 The whisker sensor.

Providing that contact between the whisker and the sensed object occurs at the whisker tip then the location of one point on the surface of the object can be found in world coordinates.

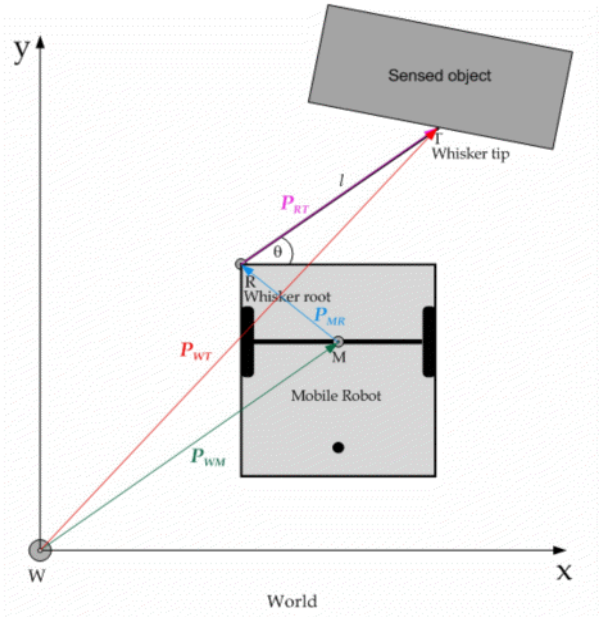


Figure 2 WhiskerBOT coordinate transformations.

Referring to Figure 2 the origin of world coordinates is W. Using odometry information WhiskerBOT maintains an estimate of the position of the turning centre of the robot M and therefore vector P_{WM} can be calculated. Taking into account the orientation of the robot, vector P_{MR} to the root of the whisker is obtained. After calibration the whisker sensor potentiometer voltage can be used to find θ and hence vector P_{RT}. Adding these three vectors gives the vector P_{WT} that locates the whisker point of contact with respect to the origin of world coordinates. WhiskerBOT is equipped with an array of eight whiskers and the aim of this project is to demonstrate that sequences of surface points recorded from these whiskers can be used to recognise and locate objects.

3 WhiskerBOT

WhiskerBOT is an enhanced version of the robot described in [Wijaya and Russell, 2002]. Two side-by-side wheels driven by stepper motors via 1250 to 1 reduction gearboxes propel the robot. A ball transfer (a captive ball bearing) provided a third point of contact with the ground. To provide the ability to grasp and transport a limited range of objects WhiskerBOT was provided with a simple 1-degree of freedom manipulator arm. The use of a four-bar linkage ensures that the attitude of a grasped object remains unchanged as it is lifted from the floor. Hitech HS425 radiocontrol servos are used to raise and lower the arm and to control the parallel finger gripper. A major upgrade to the original WhiskerBOT involved the installation of an array of eight whisker sensors.

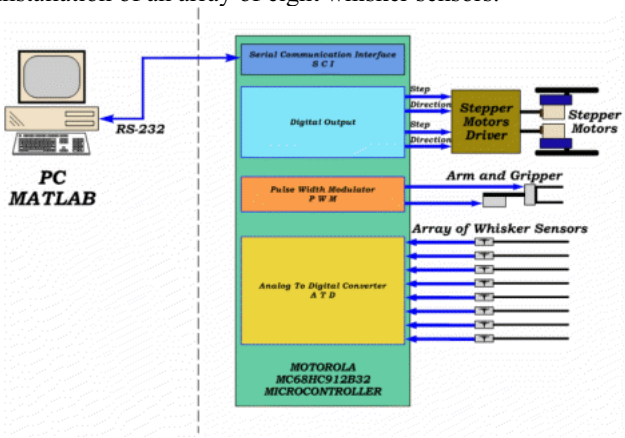


Figure 3 A block diagram of WhiskerBOT's electronics.

All sensing and actuation functions of WhiskerBOT are controlled by a 68HC912B32 microcontroller (Figure 3). The microcontroller generates pulses to activate the wheel stepper motors and arm servos. It also reads the rotation angle of each whisker potentiometer via 8 ADC channels. Analysis of sensor data and planning of the robot's actions take place on a remote PC running Matlab. Figure 4 gives an overview of WhiskerBOT.

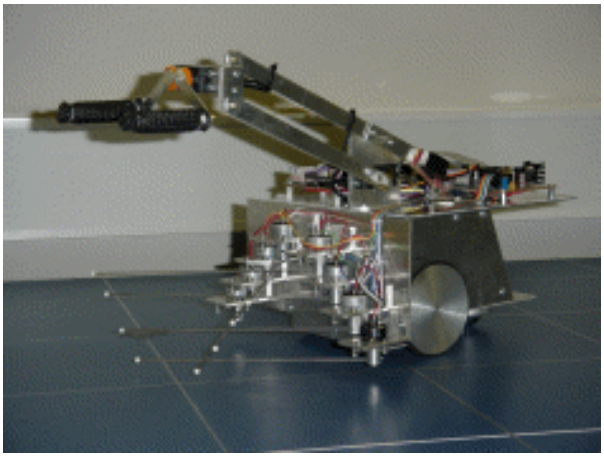


Figure 4 WhiskerBOT showing the whisker array and

simple gripper.

4 The Test Objects

In order to constrain the object recognition problem to manageable proportions a small set of convex objects was selected. These objects, shown in Figure 5 are made up of plane, cylindrical and spherical surfaces. Of these objects the tetrahedron is the only shape that could not be picked up by WhiskerBOT's simple gripper. However, the larger cylinder (a Powerade canister) was also too large for the gripper. It is further assumed that each object will be standing on one of its plane surfaces (except for the sphere) in a substantially flat environment.

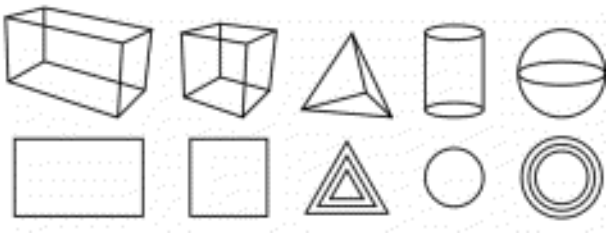


Figure 5 The five experimental shapes.

When any of the experimental objects is scanned by a whisker sensor the resulting sequence of points will either form a straight line or a convex circular arc. For this reason, techniques for fitting lines and circular arcs to groups of sensor points are required as a basic building blocks of the recognition process.

5 Fitting Geometric Primitives

The projection of the cross-section of objects considered in this project onto the *x*-*y* plane can be decomposed into two simple geometric primitives, line segments and circular arcs. Therefore, a technique is required to approximate sequences of surface points by a line segment or a circular arc. Fitting geometric primitives can be used to cluster the data points to one primitive (line or circle). It can also be used to extract important features from the data. The data points are a set of measurements of whisker tip position as it slides over the surface of the object. Thus, it can be assumed that these data correspond to the touched object, whether they represent surface points or just tip positions resulted from contact along the length of the whisker.

The following algorithms attempt to fit the two possible primitives to the data points and selects the primitive that can be fitted with the minimum mean square error. However, a line can also be represented by a circle with a large radius of curvature, therefore the selection of the primitive also depends on the radius of curvature of the circle. Only circles with radii below a certain threshold are considered valid.

5.1 Line Fitting

A least squares line fitting method was used in this project. This method is based on minimising the sum-of-squares of the perpendicular distance from the line to the data points [Weisstein, 2003].

Given n points (x_i,y_i), $1 < i < n$, the sum-of-squares of the residuals of the best-fit line is defined by:

$$F = \sum_{i=1}^{n} d_i^2 \tag{1}$$

where d_i is the perpendicular distance from the point (x_i,y_i) to the line. If the line satisfies the equation:

$$y = ax + b \tag{2}$$

where a and b are the line parameters, then:

$$d_i = \frac{|\, y_i - (ax_i + b)\,|}{\sqrt{1+a^2}} \tag{3}$$

thus, the objective function to be minimised is:

$$F(a,b) = \sum_{i=1}^{n} \frac{[y_i - (ax_i + b)]^2}{1+a^2} \tag{4}$$

The minimum of $F(a,b)$ is obtain when $\frac{\partial F}{\partial a} = 0$ and $\frac{\partial F}{\partial b} = 0$. Solving the partial differential equations of these functions with respect to a and b gives the line parameters of::

$$a = -B \pm \sqrt{B^2 + 1} \tag{5}$$

and

$$b = \bar{y} - a\bar{x} \tag{6}$$

where

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{7}$$

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{8}$$

and

$$B = \frac{1}{2}\frac{(\sum_{i=1}^{n} y_i^2 - n\bar{y}^2) - (\sum_{i=1}^{n} x_i^2 - n\bar{x}^2)}{n\bar{x}\bar{y} - \sum_{i=1}^{n} x_i y_i} \tag{9}$$

Equation 5 gives two roots of a. The intercept **b** is found by substituting a into equation 6. The correct pair of line parameters a and b is the one that gives a minimum value of the sum-of-squares of the residuals after substituting a and b into Equation 4.

For evaluation purpose, the mean square distance error of the data points to the line can be calculated using:

$$e_{line} = \frac{1}{n}\sum_{i=1}^{n} \frac{[y_i - (ax_i + b)]^2}{1+a^2} \tag{10}$$

5.2 Circle Fitting

The circle fitting method is based on minimising the sum-of-squares distance from the circle to the data points. The

circle is parameterised by its centre (a,b) and its radius r. Given n points (x_i,y_i), $1 < i < n$, the sum-of-squares of the residuals of the best-fit circle is defined by:

$$F = \sum_{i=1}^{n} d_i^2 \tag{11}$$

where d_i is the geometric distance from the point (x_i,y_i) to the circle. If the circle satisfies the equation:

$$(x-a)^2 + (y-b)^2 = r^2 \tag{12}$$

then:

$$d_i = \sqrt{(x_i-a)^2 + (y_i-b)^2} - r \tag{13}$$

thus, the objective function to be minimised is:

$$F(a,b,r) = \sum_{i=1}^{n} \left[\sqrt{(x_i-a)^2 + (y_i-b)^2} - r \right]^2 \tag{14}$$

The minimisation of Equation 14 is a nonlinear problem that cannot be accomplished by a finite algorithm. In this project, the Levenberg-Marquardt optimisation algorithm is used to solve for the circle parameters (a,b) and r. The Matlab **lsqnonlin** function in Optimisation Toolbox is used for the implementation of the Levenberg-Marquardt algorithm. The Levenberg-Marquardt algorithm is an iterative algorithm that can solve any nonlinear least squares problem provided the first derivatives of the objective function with respect to the parameters can be computed. A rapid convergence of this algorithm depends on an accurate initial guess of the circle parameters, denoted as (a_0,b_0) and r_0. An initial guess is provided by a fast and non-iterative procedure called algebraic fit.

In this method, the sum of squares of the algebraic distances between circle and data points is minimised instead of minimising the sum of squares of the geometric distances. The algebraic distance to a circle is given by:

$$\begin{aligned} F_0(a_0,b_0,r_0) &= \sum_{i=1}^{n} \left[(x_i-a_0)^2 + (y_i-b_0)^2 - r_0^2 \right]^2 \\ &= \sum_{i=1}^{n} \left[(x_i^2+y_i^2) + Ax_i + By_i + C \right]^2 \end{aligned} \tag{15}$$

where A=-2a_0, B=-2b_0, and $C = a_0^2 + b_0^2 - r_0^2$.
The minimum of $F_0(a_0,b_0,r_0)$ is obtain by simultaneously solving:

$$\begin{aligned} \frac{\partial F}{\partial A} &= \sum_{i=1}^{n} \left[x_i(x_i^2+y_i^2) + Ax_i^2 + Bx_iy_i + Cx_i \right] = 0 \\ \frac{\partial F}{\partial B} &= \sum_{i=1}^{n} \left[y_i(x_i^2+y_i^2) + Ax_iy_i + By_i^2 + Cy_i \right] = 0 \\ \frac{\partial F}{\partial C} &= \sum_{i=1}^{n} \left[(x_i^2+y_i^2) + Ax_i + By_i + C \right] = 0 \end{aligned} \tag{16}$$

Solving this system gives A,B,C, and finally the initial guess of circle parameters (a_0,b_0) and r_0 can be calculated.

These initial guesses of circle parameters (a_0,b_0) and r_0 are then used as the initial values for the minimisation of Equation 14.

In a similar method line fitting, the mean square distance error of the data points to the circle can be calculated using:

$$e_{circle} = \frac{1}{n} \sum_{i=1}^{n} \left[\sqrt{(x_i-a)^2 + (y_i-b)^2} - r \right]^2 \tag{17}$$

5.3 Contact Classification

As the robot moves, contact between the whisker and an object can occur in the following ways:

1) The tip of the whisker is always touching the surface of the object;
2) Initially the tip of the whisker is touching the surface of the object and then the tip of the whisker leaves the surface of the object;
3) The whisker is always touching the surface of the object along the length of the whisker (not at the tip);

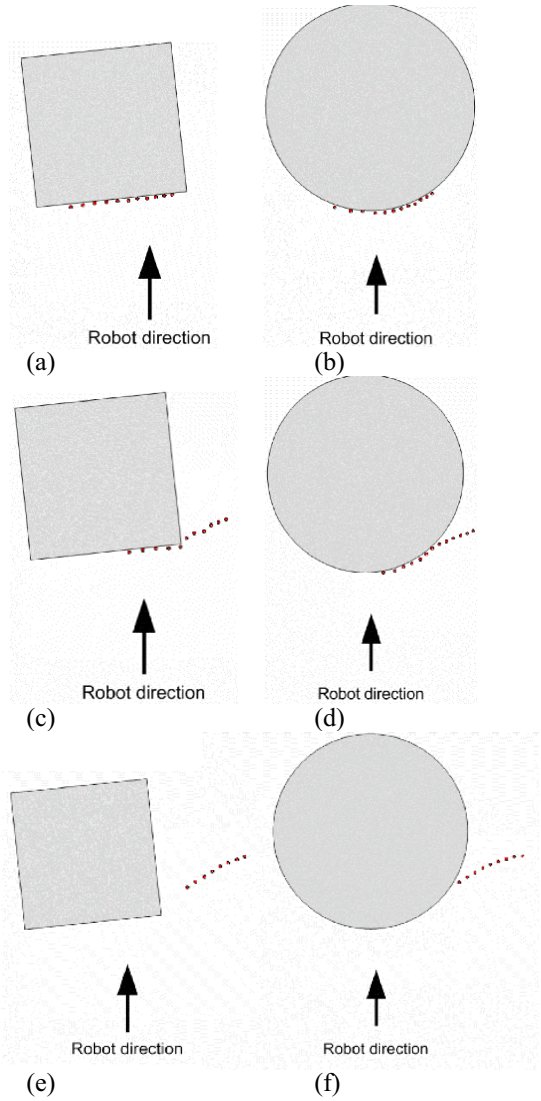


Figure 6 Whisker tip positions resulting from different forms of contact.

Figure 6 shows all of the possible contacts that may occur between the whisker and the object. Given the two types of object cross-section which consist of lines and circles, the trajectory of the whisker tip can be classify into:

- line segment,
- line segment, corner then concave circular arc
- convex circular arc,
- convex circular arc then concave circular arc, and
- concave circular arc

If these features can be successfully extracted then they lead to classification of contact (whether it is a tip contact or a contact along the length of the whisker). Simultaneously, classification of the shape of the object cross-section is also produced.

The positions of the whisker tips are stored in an ordered array. Thus, each point must relate to the object being touched, whether it is a tip contact or a contact along the length of the whisker. Each point in the array is also related to the previous adjacent point. Classification of contact, whether it is a tip contact or a contact along the length of the whisker, can be done by detecting features, such as where the corner occurred and where the whisker tip trajectory changes its curvature. All points before these features are considered tip contacts, and all points after these features are contacts along the length of the whisker and can be discarded.

5.4 Line and Corner Detection

Line detection and corner detection can be performed simultaneously. In Figure 6(a), a line occurs when the whisker tip slides over the surface of a plane object. A corner is a special case and only occurs when the whisker tip initially touches the surface of a plane object, and later lifts off the surface (Figure 6(c)). From this observation, it is crucial to detect the corner first. Once the corner has been located, the previous points can then be fitted to a line. Finally, an estimate of the corner point is produced.

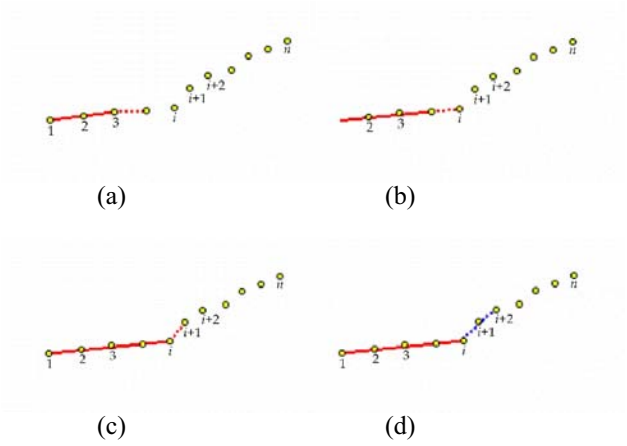


Figure 7 Algorithm to detect lines and corners. The line is successfully fitted to the first five points. The 6th and 7th points give a large angle of deviation, and thus the algorithm terminates.

Figure 7 shows how the algorithm works. The algorithm starts by fitting a line ($line_1$) to the first two points (i=1 and i=2). It then fits another line ($line_2$) to the last point (i=2) used to calculate $line_1$ and the next consecutive point (i=3). If the angle between $line_1$ and $line_2$ is less than a certain threshold, the point (i=3) is added to $line_1$ and this line is updated as the current line. The algorithm then continues to the next point until a large angle difference is detected or there are no more points left. If the angle between those two lines is larger than a threshold, it is assumed that a corner has been detected.

Noisy data can cause the algorithm to fail to detect the true corner. A noisy data point can give a large angle between the two lines. Thus the algorithm would interpret this as a corner. To overcome this problem, another check is performed. This check involves fitting a new line to at least three points, i to i+2 (given there is enough points left in the array), starting from the last point in the current line and check the angle between this new line and the current line. Again, if this angle is less than a certain threshold, the algorithm continues, otherwise, the algorithm terminates and returns the current line together with the data points as the surface points. The corner point is estimated as the intersection point of the current line and the whisker line passing through the whisker root and the whisker tip. This whisker line corresponds to the point that gave the large change of angle.

When there is a corner in the data points, the algorithm is able to detect it, and return the surface points lying on the planer surface together with the corner point. However, when no corner is detected, the algorithm will return all of the points, whether the feature is a line segment or a circular arc. In the case of a circular arc, the angle slowly changes in small increment, and thus the line-fitting algorithm fails. When all the data points are returned, the algorithm will then fit a circle to the data points and check the radius of the circle. If the radius is larger than a certain threshold, a line is fitted instead. Otherwise, the algorithm assumes it is a circular arc and the next algorithm is performed to check whether the data corresponds to a convex or concave surface.

5.5 Circular Arc Detection

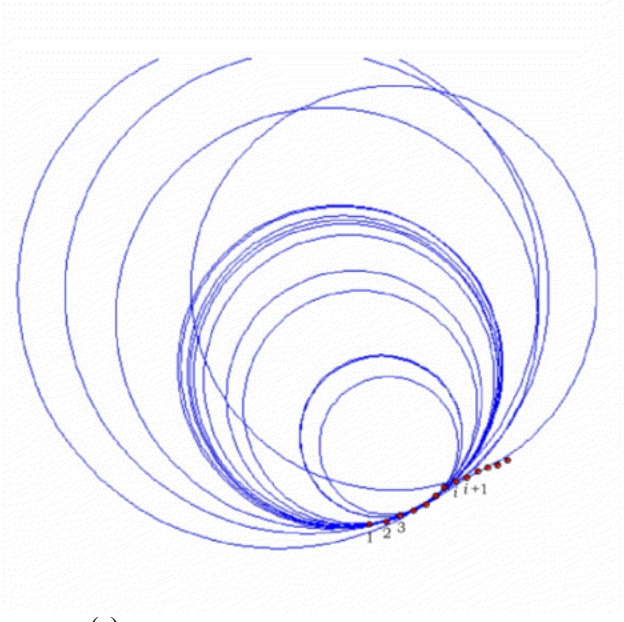
If the data points do not fit a line, the algorithm attempts to fit them to a circle instead. The circle-fitting algorithm used in this project is very fast and robust because of the small number of data points. Thus, recursively fitting a circle to a series of data points can be performed without a large computational burden.

The trajectory of the whisker tip can follow the shape of a circular arc in two ways: when the whisker tip slides over the surface of a circular object (Figure 6(b)) and when the whisker touches an object along the length of the whisker (Figure 6(e) and Figure 6(f)). If the whisker touches the surface of a circular object, such as cylinder or sphere, the trajectory of the whisker tip will form a convex circular arc with respect to the whisker root. And, if the whisker touches an object along the length of the whisker, the trajectory of the whisker tip will also form a concave circular arc with respect to the whisker root. Both convex and concave circular arcs occur when the whisker tip initially touches the surface of a circular object and is then lifted off the surface of the object (Figure 6(d)).

Valid circular arcs are those formed by contacts between the whisker tip and the surface of a circular object. Extracting a valid circular arc can be done by fitting a circle to the data points and evaluating the radius

and the centre location of the circle. A valid circular arc has a radius below a certain threshold and a convex shape with respect to the whisker root.

An algorithm has been developed to automatically classify points corresponding to tip contact and those that do not. This algorithm works by recursively fitting a circle to a series of data points and voting for the points that return a valid circle (based on the radius and the centre location).



(a)

(b)

Figure 8 Algorithm to detect a circular arc. (a) All valid circles are plotted, and (b) For each surface point the vote histogram shows how frequently they receive votes.

Figure 8 illustrates the working of the algorithm. For all data points, start by fitting a circle to a sequence of four data points starting from the first point in the array. Check whether it is a valid circle or not. If it is a valid circle, increment the vote for all of the corresponding points. Then, get a new sequence of four data points starting from the second point, and repeat until all data points has been voted. Next, increment the number of data points in the sequence and repeat the whole process until the number of data points in the sequence equals the number of data points in the array.

The classification of surface points is based on the number of votes for the point using the knowledge that tip contacts most probably occur in the first few data points in the array. Hence, starting from the second point, if the votes for that point is equal or greater than the votes for the first point, then add that point to the array of surface points, otherwise terminate and return the array of surface points. The rest of the points that are not added to the array of surface points can be discarded.

This algorithm works well even with the presence of noise in the data. It is very rare that the first few points, if they really are tip contacts, give an incorrect vote. Once the surface points have been extracted, a circle can then be fitted to these points.

If the algorithm returns all the data points as a concave circular arc (i.e. no votes for valid circle at all), then there are two possibilities. The object may really have a face with the shape of a concave circular profile or the shape is the result of contact along the length of the whisker. To verified this, the tip test algorithm is performed [Wijaya and Russell, 2002]. However, instead of checking each point, the algorithm picks only the middle and the first points of the sequence of the fitted circular arc, and tests these points using the tip test algorithm. This way, the execution time and the errors from backlash and wheel slippage are minimised.

5.6 Plane, Cylinder and Sphere Fitting

All data points corresponding to extracted line segments are projected onto the *x*-*y* plane and those that form line segments with a similar orientation are fitted to a single plane. A least squares plane-fitting method is used that is an extension of the line fitting technique described in Section 5.1.

Since the cylindrical objects used in this project are resting on one of their planer surfaces, the model of the cylinder can be obtained by using the projection of the cylinder onto the *x*-*y* plane, which is a circle. All the data points from the extracted circles from each whisker that have a similar radius are fitted to a single circle using the methods described in Section 5.2. Then, the cylinder is modelled by a circle of centre (a,b), radius r, and height equal to the maximum z coordinate of any whisker that touches the object.

If the data points from each whisker form concentric circles of different radii when projected onto the *x*-*y* plane then they are fitted to a sphere. The sphere was parameterised by its centre (a,b,c) and its radius r. The parameters were estimated by minimising the sum-of-squares distance from the sphere to the data points using an extension of the circle fitting technique described in Section 5.2.

6 Object Recognition

Object recognition involved traversing the interpretation tree shown in Figure 9. Information about the surface type, size and orientation was used to discriminate between the shapes of the test objects. Tolerances were placed on the values of cylinder radius, face width and face dihedral angle. These tolerances were used to discriminate between the test objects and to identify as 'unknown' objects that returned values that did not correspond to any of the test objects. Once an object was recognised a model of the object was fitted to the sensor data as illustrated in Figures 10-13.

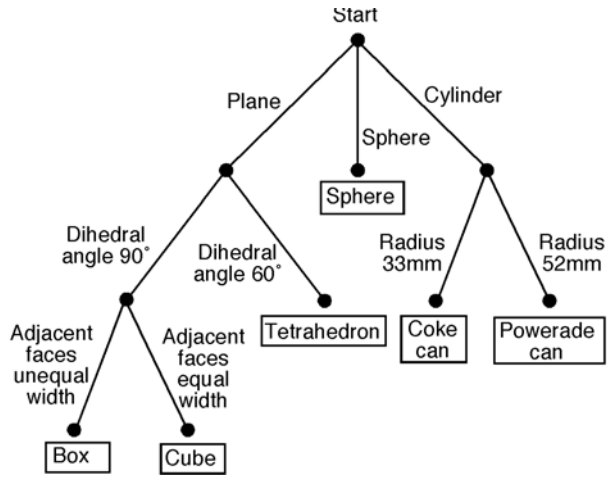


Figure 9 The interpretation tree for the test objects.

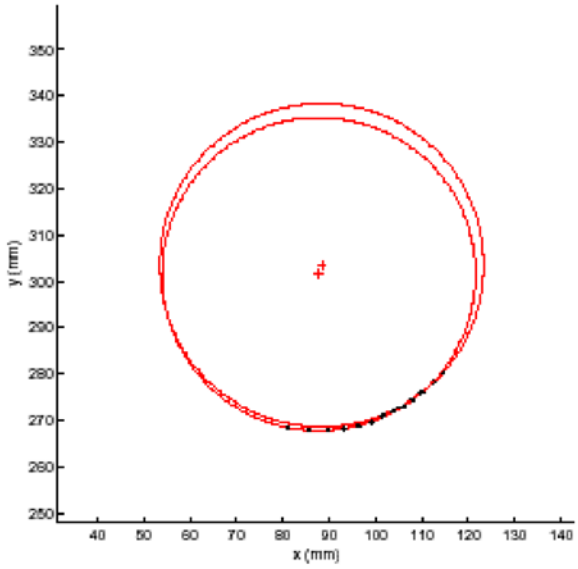


Figure 10 Projection of the fitted circles onto the *x*-*y* plane for the Coke can.

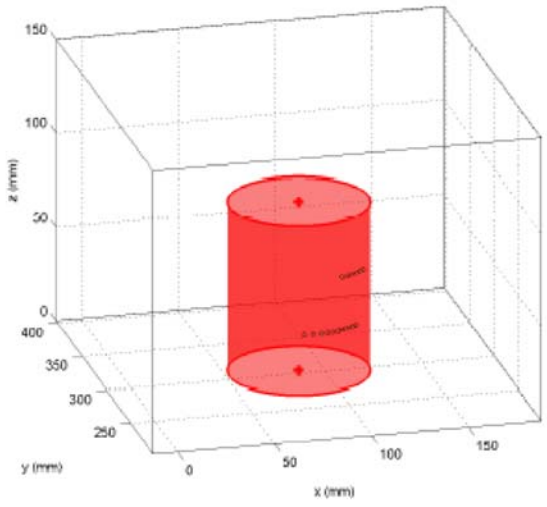


Figure 11 The outline of the Coke can fitted to the measured surface points.

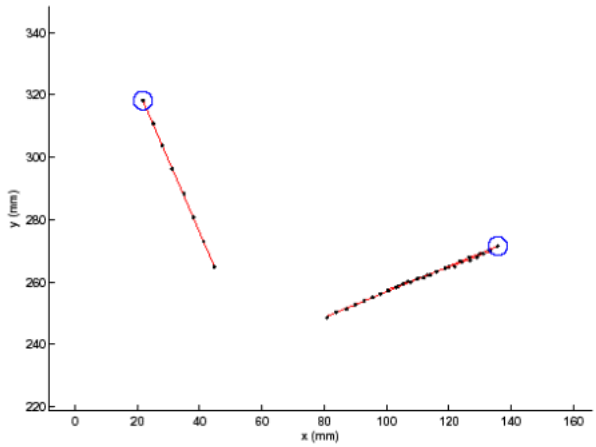


Figure 12 Projection of the fitted lines and corners (indicated by circles) onto the *x*-*y* plane for the box.

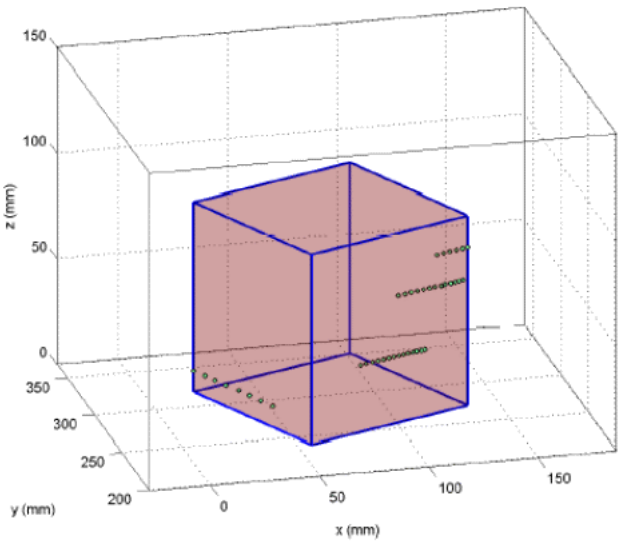
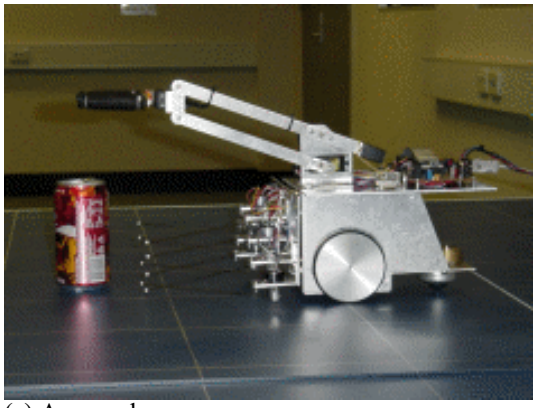


Figure 13 The outline of the box fitted to the measured surface points.

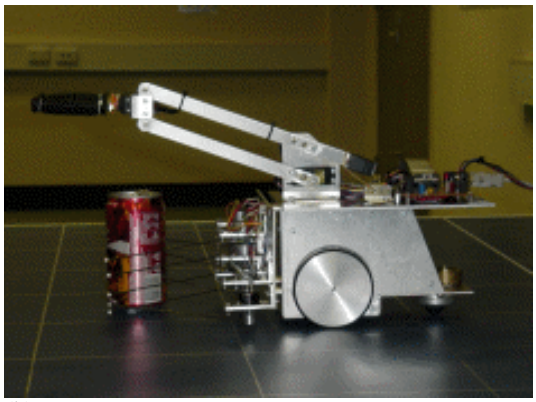
7 Object Identification and Retrieval

As a demonstration of the capabilities of WhiskerBOT an object retrieval task was programmed. Starting from a 'home' position WhiskerBOT searched a tabletop environment to find a specific object in its list of test objects. Because WhiskerBOT lacks any long-range sensors, a raster-scan search was employed. The separation between successive scans was made to be the same as the span of the whisker sensor array. Upon detecting an object the whiskers were scanned over the object to gather surface information. Using this surface information the object was recognised and located. If the object corresponded to the one being sought then WhiskerBOT picked it up and returned it to the 'home' position.

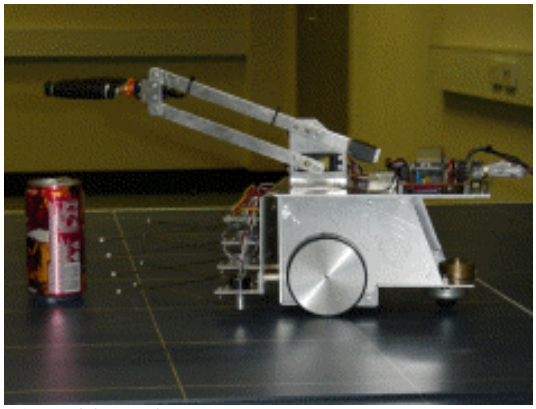
Figure 14 shows images taken from a video showing WhiskerBOT successfully recognising and retrieving a soft drink can.



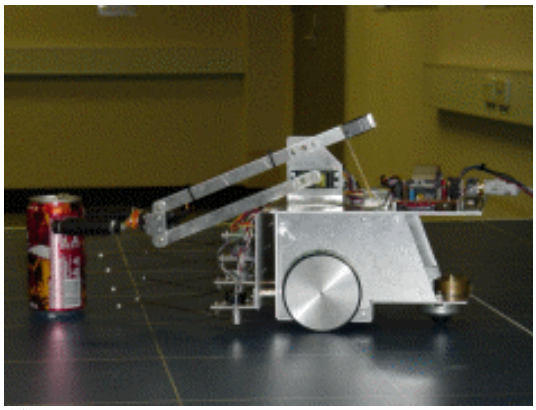
(a) Approach



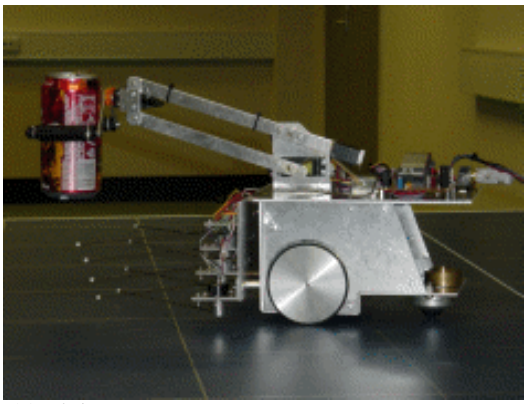
(b) Sense



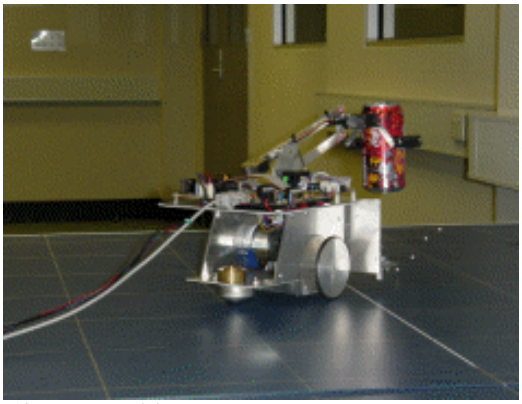
(c) Position gripper



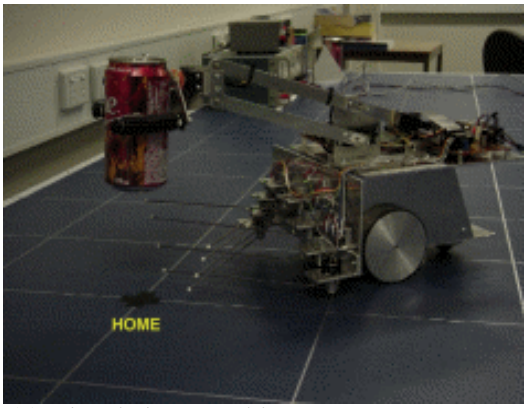
(d) Grasp



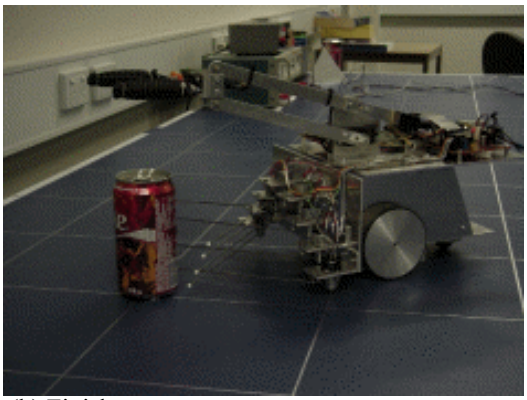
(e) Pick up



(f) Return with can



(g) Place in home position



(h) Finish

Figure 14 Retrieving a soft-drink can.

8 Conclusions

In this project, a whisker sensor based mobile robot, called WhiskerBOT, has been developed. The robot has been provided with accurate odometry and carries an array of eight whiskers that are used as its only sensory modality. Each whisker is mounted on a potentiometer to measure deflection caused by contact with external objects. The whiskers are passive and rely upon the motion of the robot in order to scan the surface profile of touched objects. WhiskerBOT is being developed to demonstrate the sensing capabilities of whisker sensors. The robot is able to recognize a few objects formed from plane, cylindrical and spherical surfaces. By using its simple manipulator, it can pick up and retrieve small objects.

Computer vision, laser rangefinders and ultrasonic sensors are common forms of robotic sensing for both terrestrial and underwater environments. However, in difficult conditions caused by dust, poor illumination, suspended particles, mud, and salinity/temperature gradients these sensory modes may be unable to function efficiently. Under these conditions tactile whiskers may be a better solution for providing close proximity sensing. This project has demonstrated that whiskers are capable of providing more than the simple contact/no contact information commonly associated with robotic whisker sensors. By using more powerful computers to process the tactile information and higher performance actuation for the mobile robot it is expected that the speed of whisker-based tactile sensing can be made comparable with computer vision and ultrasonic sensing systems.

References

[Layhousen, 1979] P. Leyhousen, *Cat Behavior*, Garland STMP Press, New York, 1979.

[McKerrow, 1990] P. McKerrow, *Introduction to Robotics*, Addison-Wesley, 1990.

[Brooks, 1989] R.A. Brooks, 'A Robot that Walks; Emergent Behaviors from a Carefully Evolved Network', *Neural Computation*, Vol 1, pp 253-262, 1989.

[Dibley, 1984] Alan Dibley DIY about the mouse, *Practical Computing*, pp. 120-121, July, 1984

[Hirose, et al., 1985] S. Hirose, et al., 'Titan III: A Quadruped Walking Vehicle', *Robotics Research - The Second International Symposium*, The MIT Press, Cambridge, Massachusetts, 1985.

[Schiebel, et al., 1986] E.N. Schiebel, H.R. Busby, and K.J. Waldron, 'Design of a mechanical proximity sensor', *Robotica*, Vol. 4, pp 221-7, 1986.

[Jung and Zelinski, 1996] David Jung and Alexander Zelinski, Whisker-based mobile robot navigation, In Proceedings of the IEEE/RJS International Conference on Intelligent Robots and Systems (IROS), Vol. 2, pp. 444-449, 4-8 November, 1996.

[Wijaya and Russell, 2002] Jaury A. Wijaya and R. Andrew Russell. Object exploration using whisker sensors. *Proceedings of the Australasian Conference on Robotics and Automation*, pages 180--185, Auckland, 27-29 November 2002.

[Weisstein, 2003] Eric W. Weisstein. Least squates fitting - perpendicular offsets. Online: http://mathworld.wolfram.com/LeastSquaresFittingPerpendicularOffsets.html last accessed 18/06/2003.