RANGE AND POSE ESTIMATION FOR VISUAL SERVOING OF A MOBILE ROBOT

David Jung, Jochen Heinzmann and Alexander Zelinsky
Robotic Systems Laboratory,
Department of Systems Engineering
Research School of Information Sciences and Engineering,
The Australian National University,
Canberra, ACT 0200, Australia
http://wwwsyseng.anu.edu.au/rsl/

Abstract

This paper describes the implementation of behaviour for real-time visual servoing on a mobile robot. The behaviour is a component of a multi-robot cleaning system developed in the context of our investigation into architectures for cooperative systems. An important feature for support of cooperation is the awareness of one robot by another, which this behaviour realises. Robust feature tracking aided by a hardware vision system is described. This forms the basis for range and pose estimation using a 3D projective model.

1. INTRODUCTION

As part of our research into behaviour architectures for cooperative multi-robot systems, we are constructing a cleaning system using two autonomous mobile robots [Jung 97]. There are numerous component behaviours implemented to support the basic cleaning task. Among them is the requirement for one robot to be aware of the other. This is realised by visual tracking, and estimating the range and pose, of the robot in its field of view in real-time. This paper briefly describes the cleaning task, but is primarily a description of the implementation of the visual servoing behaviour.

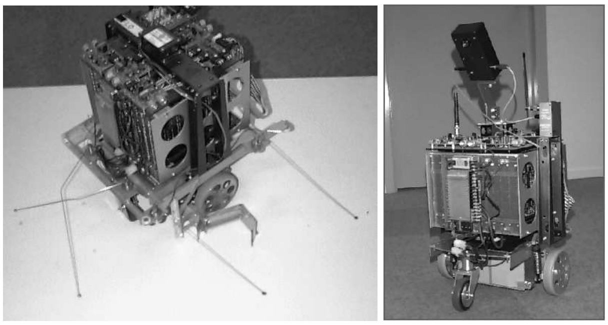

Figure 1 - Yamabicos Flo and Joh

2. COOPERATION

Research into multi-robot systems is driven by the assumption that multiple agents have the possibility to solve problems more efficiently than a single agent does. Agents must therefore cooperate in some way. There are many tasks for which a single complex robot could be engineered; however, in many cases there are advantages to using multiple robots. A multi-robot system can be more robust because the failure of a single robot may only cause partial degradation of task performance. In addition, the robots can be less complex since each is only responsible for partial fulfilment of the task. Our philosophy is to design heterogeneous multi-robot systems where appropriate.

Cooperation has been intensively studied in the robotics community recently. Much attention has been given to the cooperation observed in eusocial insect societies, such as ants (see [Arkin 92b][Kube 94] [Mataric 95][Steels 90]). This type of cooperation has been termed *collective robotics* and is characterised by homogeneity of the agents, genetically determined behaviour (designed robot cooperation), and anonymous societies. In this case the cooperation is said to be *emergent*. Essentially the social system is a super-organism that controls its components, which is similar to how collections of cells make organs and collections of organs make organisms. The robotics literature usually reserves the term cooperation for cases where explicit communication is involved.

The next level of sophistication of social organisation in biological systems requires autobiographical agents. In this case individuals can identify each other in the society. Each individual maintains a representation of its historical experience, which is a large part of its identity.

The effect of agent identity and capability for sophisticated communication greatly increases the possibilities for types of cooperative behaviour. Much research has been reported where robots communicate. Some systems utilise central control, others use complex negotiation schemes between agents, and yet others some combination of both [Kuniyoshi 94][Noreilis 92] [Parker 94][Yuta 90].

The behaviour architecture we have developed for distributed planning, ABBA, supports all the above mentioned paradigms [Jung 97b]. However, we are particularly interested in heterogeneous agents that have identity, the capability to communicate, and can jointly plan cooperative behaviour without central control. Since the robots have identity, awareness of one robot by another can play a major role in these cooperative systems, and has been investigated in the literature [Parker 95]. This provides the motivation for implementing the capability for one robot to visually detect, recognise, track, and servo on another robot. The task we have implemented utilises this behaviour by design.

The remainder of this paper discusses the implementation of this visual behaviour, after introducing the task context.

2.1 Multi-robot Cleaning

The concrete task we chose to implement for our investigation into cooperative systems is for our two autonomous mobile robots to clean the floor of our laboratory. The 'Yamabico' robots [Yuta 91] shown in Figure 1 are heterogeneous in the sense that each has different tools and sensors such that neither can accomplish the task alone.

One of the robots, *'Joh'*, has a vacuum cleaner that can be turned on and off via software. Joh's task is to vacuum piles of litter from the laboratory floor. It cannot vacuum close to walls or furniture. It has the capability to 'see' piles of litter using a CCD camera and a video transmitter that sends video to the *Fujitsu MEP tracking vision system*. The vision system is capable of landmark-based navigation and can operate safely in dynamic environments at speeds up to 600mm/sec [Cheng 96]. The vision system can communicate with the robot, via a UNIX host, over a pair of radio modems.

The other robot *'Flo'*, has a brush tool that is dragged over the floor to sweep distributed litter into larger piles for Joh to pick up. It navigates around the perimeter of the laboratory where Joh cannot vacuum and deposits the litter on open floor space. It's primary sensing is via four proportional whisker sensors developed specifically for wall following and perimeter navigation [Jung 96a].

The task is to be performed in a real laboratory environment. Our laboratory is cluttered and the robots have to contend with furniture, other robots, people, opening doors, changing lighting conditions, equipment with dangling cables and other hazards.

3. VISUAL TRACKING

One implementation of the cleaning task involves Joh waiting until Flo is in its field of view, and then visually servoing on Flo until Flo dumps a pile of litter in the vicinity. This requires not only robust tracking for servoing while Flo is in view, but also the ability to continuously determine if Flo in the image. This section details how this is achieved. First, a look at our vision system.

3.1 The Fujitsu vision system

The Fujitsu MEP template tracking vision system consists of two VME bus cards – a video module and a tracking module. Both are accessed over the back plane using a processing card containing a 68040 CPU running at 25Mhz. The tracking module contains a correlation chip that is used to perform correlation calculations between stored templates and the NTSC video stream at 30Hz. The correlation processing occurs after the video module has digitised a frame, and hence the results are always a frame behind the video stream.

Before processing starts the CPU provides a set of, up to 100, template addresses to be correlated in the coming frame. The templates can be 8 or 16 pixels high or wide, and are all 8bit grey-scale. With each template a search area in the image is specified, as the system cannot correlate against the entire image for each template. The search areas are divided into a 16×16 grid since the tracking module performs 256 correlations per template in each search area. The tracking module performs a correlation calculation between a template and all possible offsets that move it within its search area. The system also supports horizontal and vertical magnification in templates, so that the template pixels are compared with every n^{th} image pixel – where n is the x or y magnification (m_x or m_y). If o_x and o_y are the offsets of the template in the image, and g_t and g_f are the template and images respectively, then the correlation calculation is simply the sum of pixel differences as given by (1).

$$D = \sum_{x=0}^{Size}\sum_{y=0}^{Size}\left| g_t(x_t, y_t) - g_f(x_f, y_f) \right| \qquad (1)$$

Where $x_t = x \cdot m_x \qquad y_t = y \cdot m_y$

$$x_f = x \cdot m_x + o_x \qquad y_f = y \cdot m_y + o_y$$

The result from the processing is a vector for each template that corresponds to the offset from the search area centre where the best correlation between the template and image was found.

By moving the centre of the search area based on these result vectors a template can be crudely tracked. Any change in brightness, occlusion, or change in size – caused by changes in range, will result in loss of tracking.

3.2 What do we track?

Since the target of our tracking is artificial – Flo, we have control over its appearance. Hence, we chose to put a specific geometric pattern on both sides of the robot, as shown in the figure below (the thick black rectangle is the pattern, the other lines are graphics).

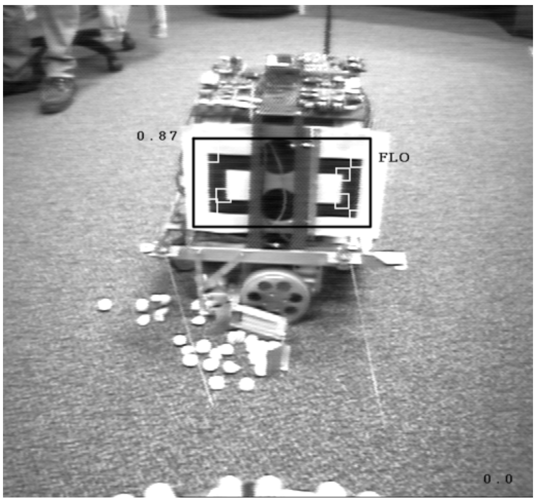

Figure 2 - Flo's side as seen from Joh's CCD camera

The pattern is a hollow rectangle with very thick sides, designed such that templates of the inner and outer corners will be scale invariant. Given the fixed 16×16-pixel size of the templates, if the robot is too far away the templates will cover more than the corners of the pattern. The size of the pattern and the templates were chosen to minimise this problem.

Another problem that arose was due to the motion involved. Because the video signal is interlaced, motion causes one field to be displaced with respect to the other. This gives very bad correlation with the template and tracking is lost. The solution was to use templates with a magnification of two, so that only every other vertical line is used for matching.

The task dictates that Flo will almost always been seen from the side. Specifically Flo spends most of its time sweeping against walls, which Joh's behaviour keeps it away from. It is sufficient for Joh to detect this pattern to 'see' Flo. Using this method we want something that detects the robot irrespective of the range or pose of the pattern.

3.3 Utilising geometric constraints

Tracking the ten templates independently – eight of the inner and outer corners, as well as two side templates, is not sufficient for tracking. The templates are quickly lost by the tracking system due to variations in brightness and changes in shape due to viewing the pattern at an oblique angle. The tracking can be made robust by utilising the geometric constraints we have between the template positions.

Our first implementation modelled the pattern as a 2D rectangle in which the size was allowed to vary. Hence, if we know the position of any two corners, the positions of the other templates are also known. Unfortunately, because the tracking is noisy we can never know for certain if we have the coordinates of any particular corner. However, we do have a measure of the uncertainty of any particular match – the correlation value of the best match returned by the tracking module. The method developed to use the noisy tracking positions with their correlation values, and the predictions from the 2D model, was a network of independent Kalman filters. The scheme was first developed in our laboratory to track features on a human face for gesture recognition [Heinzmann 97].

The Kalman filter is a recursive linear estimator, which merges two independent measurements of a quantity with known measurement error to give the optimal estimate of the value and associated error [Boznic 86][Maybeck 79]. The use of Kalman filtering in feature tracking has previously been reported by McLaughlan *et al.* [McLauchlan 94]. Our approach also uses a Kalman filter for each feature, but it also uses the information from all other features.

The Kalman filter takes two measurements of a feature position and their associated errors as input, and gives a new optimal estimate of the position. The first measurement comes from the vision system tracking module. If we let $fp_i(t-1)$ denote the estimated feature position of feature *i* from the previous frame, and m_i the displacement from the search area centre to the new best template match for the feature, then the measurement becomes (2).

$$\overline{mp_i}(t) = \overline{fp_i}(t-1) + \overline{m_i}(t) \tag{2}$$

Now, the second 'measurement' comes from the 2D model of the rectangle. The predicted position of feature *i*, $\overline{pp_i}$ in this case, uses the displacements between features according to the model. This displacement vector between *i* and *j* is $\overline{d_{ij}}$. The prediction of the position of the feature proceeds as follows. All features are iterated over, and in each case its position in the previous frame $\overline{fp_j}(t-1)$ has the displacement between it and feature *i* subtracted from it. This gives an estimate of the position of feature *i* according to the model and the last position of feature *j*. This term is weighted by the certainty in the position of feature *j*, and the terms are summed for all *j* (including *i=j*). After re-normalising the effects of the weights, we have an estimate of the position of feature *i* according to the model and a weighted contribution of all the other features. Equation (3) shows the formulation, where *P* is the variance.

$$\overline{pp_i}(t) = \frac{\sum_j \frac{\overline{fp_j}(t-1) - \overline{d_{ij}}(t-1)}{P_j(t-1)}}{\sum_j \frac{1}{P_j(t-1)}} + \overline{m}(t) \tag{3}$$

Here $\overline{m}$ is the gross motion of the whole set of features, again weighed by the variances, from the vision module. It is added to the predicted position of the feature as a prediction of future motion. This assumes that the motion of the whole feature set – ie. the object being tracked – exhibits roughly linear motion. The acceleration effects are very small from one video frame to the next except for unrealistic accelerations. The calculation for $\overline{m}$ follows.

$$\overline{m}(t) = \frac{\sum_{i=1}^{n} \frac{\overline{m_i}(t)}{P_i(t-1)}}{\sum_{i=1}^{n} \frac{1}{P_i(t-1)}} \tag{4}$$

The predicted position variance is calculated analogously. Now we have the 'measurements' from the vision tracking module $\overline{mp_i}$ and from the model $\overline{pp_i}$ for each feature.

Next we use the Kalman filter to obtain the new estimate of the feature position.

$$\overline{fp_i}(t) = \overline{pp_i}(t) + K_i(t)\left(\overline{mp_i}(t) - \overline{pp_i}(t)\right) \tag{5}$$

All that remains is to calculate new search areas for the matching in the next frame. Then the process is iterated again. All of this calculation is performed at frame-rate on the 68040 CPU. There is also a low priority process, which sweeps search areas over the entire image looking for a better match for each feature than the current best match. This is to allow the re-acquisition of tracking if it is completely lost due to occlusion.

Using this technique with a 2D sizeable rectangular model the tracking is very robust to changes in range. It is also moderately robust to changes in the angle between rectangle and the plane of the camera. Variations in brightness caused by the automatic shutter speed adjustment on our camera can also be compensated for. We match a number of templates of various brightness at the same position for a given feature.

4. Range and Pose Estimation

To satisfy the requirements of the cleaning task Joh must be able to determine Flo's range and pose. In one experiment Joh notifies Flo when Flo has been visually acquired. Flo notes its position and may later communicate the location of a litter pile for Joh to vacuum using coordinates relative to this location. Hence Joh needs Flo's position and heading to know the coordinate reference frame. Since Joh can identify piles of litter visually once in the vicinity, high accuracy was not required.

4.1 A First Cut

Our first implementation was very simple. We observed that since the camera is inclined toward the floor, the range to Flo is proportional to the height of its pattern in the image. So a rough calibration provided a simple measure of range. We also noted that due to perspective foreshortening, one side of the rectangle becomes smaller with respect to the other, in proportion to the angle of inclination to the camera plane.

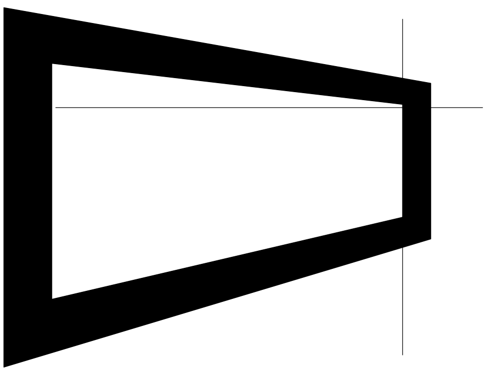

Figure 3 - Perspective foreshortening

The ratio of these two sides gives a measure of the inclination angle. Although this mechanism works, unfortunately is was not accurate enough for our needs. There are a number of problems. First, because the model used for tracking is a 2D rectangle, it is working against predicting the correct feature positions for the corners if the angle is great, as can be seen from Figure 3. Also in this situation, the image of the corners themselves is no longer a right angle. The templates used for tracking all the corners are right angled. Hence tracking is easily lost when Flo is inclined over 40^o out of the camera plane. This error has a large effect when use to project Flo's relative coordinate specifications of litter piles.

Clearly we needed a 3D model to track robustly in the situation when perspective foreshortening becomes significant. Much research has been conducted on recovering 3D-pose information given 2D projections, but in most cases, for full recovery of points, weak-perspective projection is assumed [Alter 92][Kanade 83][Grimson 92]. A weak perspective, or affine, projection is orthographic projection plus scaling. This essentially ignores perspective foreshortening by assuming all 3D points are roughly the same distance from the camera. Hence we require full perspective projection.

4.2 Inverse Projective Geometry

The 3D model includes the 3D coordinates of the outside corner points on the rectangular pattern. We need to be able to determine these 3D locations from the 2D positions in image space. We also need to be able to re-project the 3D locations onto the feature positions in image space during tracking. Hence, we need the projective transformation matrix of the camera.

A projective transformation has eight parameters. There are a number of constraints we can take advantage of. We know the absolute size of the pattern. We also know that the pattern moves only in a plane inclined to the camera. This is because as the robot moves along the floor plane, which is inclined to the camera at a fixed angle, the height of the pattern from the floor doesn't change. In addition the pattern only rotates about one axis (perpendicular to the floor) – the robot does not tilt to one side. These two constraints imply that the 3D orientation of the lines on the left and right side of the pattern doesn't change.

Calibration

We need to know the height of the robot's camera from the floor plane in which the other robot moves, and the angular tilt forward. Because this is only a translation and a rotation about one axis, the value can be recovered from the camera projection matrix. This can be obtained by a calibration procedure. Flo is placed at a known distance in front of the camera and the pattern aligned with the camera image plane. Now we know the absolute positions in 3D of the corner points, and using the tracking with a simple 2D rectangular model, we can also obtain the image projection of each corner. These four 3D points are their corresponding projections can uniquely define the matrix. If t_{ij} are the elements of the 3×3 projection matrix T, and the four corner points are $(\lambda_i x_i, \lambda_i y_i, \lambda_i)^t = T(X_i, Y_i, 1)^t$, then the system below is solved [Kapur 85].

$$\begin{bmatrix} X_1 & Y_1 & 1 & 0 & 0 & 0 & -x_1X_1 & -x_1Y_1 \\ 0 & 0 & 0 & X_1 & Y_1 & 1 & -y_1X_1 & -y_1Y_1 \\ X_2 & Y_2 & 1 & 0 & 0 & 0 & -x_2X_2 & -x_2Y_2 \\ 0 & 0 & 0 & X_2 & Y_2 & 1 & -y_2X_2 & -y_2Y_2 \\ X_3 & Y_3 & 1 & 0 & 0 & 0 & -x_3X_3 & -x_3Y_3 \\ 0 & 0 & 0 & X_3 & Y_3 & 1 & -y_3X_3 & -y_3Y_3 \\ X_4 & Y_4 & 1 & 0 & 0 & 0 & -x_4X_4 & -x_4Y_4 \\ 0 & 0 & 0 & X_4 & Y_4 & 1 & -y_4X_4 & -y_4Y_4 \end{bmatrix} \begin{bmatrix} t_{11} \\ t_{12} \\ t_{13} \\ t_{21} \\ t_{22} \\ t_{23} \\ t_{31} \\ t_{32} \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ x_4 \\ y_4 \end{bmatrix} \qquad (6)$$

Where t_{33}=1. This process is performed only once, by manually placing Flo at the required position in front of the camera. This projection matrix T is used to perform the forward projection from the 3D model points to the image space during tracking – to predict the feature locations.

From 2D to 3D

Consider the line joining the top and bottom corners of one side of the rectangular pattern. This is a line of known length, and known direction – it always runs perpendicular to the floor. Let the projected image end points of this line, which are available from the tracked feature positions, be $(u_1, v_1)^t$ and $(u_2, v_2)^t$, and the 3D line $(a,b,c)^t + \rho(m_1, m_2, m_3)^t$. Then, since we know the direction cosines $(m_1, m_2, m_3)^t$ and the length ρ, the solution for both end-points can be calculated from (7) [Haralick 93].

$$c = \frac{\rho[(u_2 - u_1)(fm_1 - u_2 m_3) + (v_2 - v_1)(fm_2 - v_2 m_3)]}{(u_2 - u_1)^2 + (v_1 - v_2)^2} \qquad (7)$$

$$a = \frac{u_1}{f} c \qquad b = \frac{v_1}{f} c$$

Where f is the known focal length of the camera.

So performing this calculation on the lines from both the left and right of the pattern gives us all the corner locations in 3D. These points are in the camera coordinate system, which has its origin at image centre and z-axis perpendicular to the image plane. So to calculate the range, bearing and heading to Flo we need to transform this to 'floor' coordinates. A translation and rotation, by amounts calculated from the camera projection matrix, easily achieve this since we know the pattern plane is perpendicular to the floor plane.

All together now

In summary, the tracked feature positions of the four corners are used in pairs in equation (7) to determine their 3D positions. These positions are passed through the camera projection matrix T and used to update the positions of the features during the next tracking frame. These same points are also transformed into 'floor' coordinates where some simple trigonometry can determine the range from Joh, the bearing angle in relation Joh, and the heading angle relative to the camera plane. This is possible because the forward tilt and height of the camera in relation to the floor can be calculated from the calibrated matrix T. These three values are then used to calculate Flo's trajectory. The trajectory is used both to visually servo on Flo by sending motion commands to Joh from the vision system via a radio modem, and also as a reference frame for litter pile location specifications communicated by Flo to Joh.

5. Experimental Results

The sequence of frames in Figure 4 was taken from video footage of Joh visually servoing on Flo while Flo comes past 'sweeping' litter from against a wall. The view is from the vision system video stream with overlay graphics. The graphics is sometimes difficult to see because of interlacing effects. The small black boxes near the corners of the pattern represent where the features are matched. The larger the box the poorer the match, so a very good match gives small spots. The frames from Figure 5 show how tracking is re-acquired after being lost due to occlusion. This video footage is available in *mpeg* format from our web page.

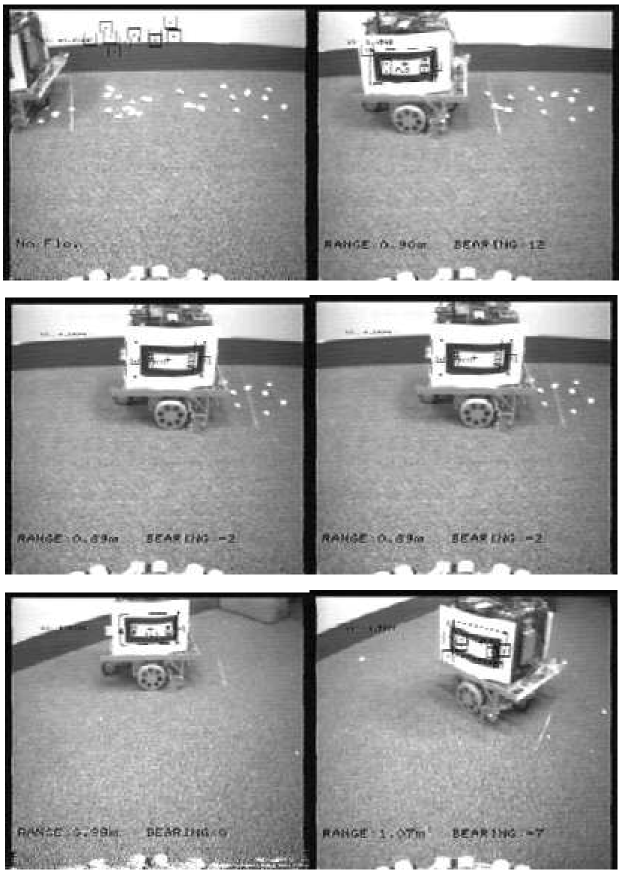

Figure 4 - Visual servoing (with range and bearing)

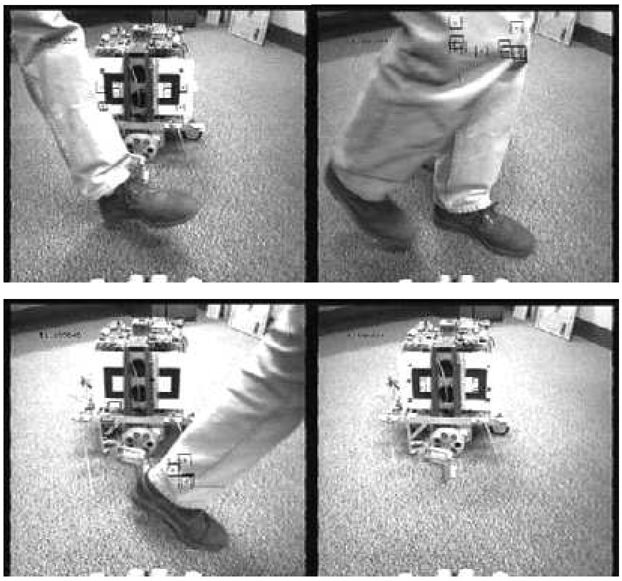

Figure 5 - Loss and re-acquisition of tracking

6. Conclusion

The method presented for tracking and calculating range, heading and bearing estimates on a known moving target provides the accuracy we require to support our cooperative behaviour. Importantly it also requires little calculation – mostly addition and multiplication, but some trigonometric functions. This allows the calculation to be performed in real-time.

This economy of calculation is largely due to the exploitation of many geometric constraints available for this particular problem.

ACKNOWLEDGMENTS

This work was supported by Fujitsu who produced our vision system, and Wind River Systems, suppliers of VxWorks. Special thanks also to Samer Abdallah for his tips on projective geometry.

REFERENCES

[Alter 92] Alter, T. D., "**3D Pose from 3 Corresponding Points under Weak-Perspective Projection**", *Massachusetts Institute of Technology, AI Laboratory*, A.I. Memo No. 1378, July 1992.

[Arkin 92b] Arkin, R. C. and Hobbs, J. D., "**Dimensions of Communication and Social Organization in Multi-Agent Robotic Systems**", *Proc. Simulation of Adaptive Behavior 92*, Honolulu, HI, Dec. 1992.

[Boznic 86] Boznic, S. M., "**Digital and Kalman Filtering**", *Edward Arnold Publishers*, 1986.

[Cheng 96] Cheng, Gordon and Zelinsky, Alexander, "**Real-Time Visual Behaviours for Navigating a mobile Robot**", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol 2. pp973. November 1996.

[Grimson 92] Grimsin, W. E. L., Huttenlocher, D. P., and Alter, T. D., "**Recognizing 3D Objects from 2D Images: An Error Analysis**", *in Proc. IEEE Conf. Computer Vision Pat. Rec.*, 1992.

[Haralick 93] Haralick, Robert M., and Shapiro, Linda G., "**Computer and Robot Vision**", *Vol. 2, Addison-Wesley Publishing Co.*, ISBN 0-201-56943-4 (v. 2), 1993.

[Heinzmann 97] Heinzmann, J. and Zelinsky, A., "**Robust Real-Time Face Tracking and Gesture Recognition**", *Proceedings of IJCAI'97, International Joint Conference on Artificial Intelligence*, August 1997.

[Jung 96a] Jung, David and Zelinsky, Alexander, "**Whisker-Based Mobile Robot Navigation**", *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol 2. pp497. November 1996.

[Jung 97] Jung, David, Cheng, Gordon and Zelinsky, Alexander, "**An Experiment in Realising Cooperation between Autonomous Mobile Robots**", *Fifth International Symposium on Experimental Robotics (ISER)*, Barcelona, Catelonia, June 1997.

[Jung 97b] Jung, David, Cheng, Gordon and Zelinsky, Alexander, "**Robot Cleaning: An Application of Distributed Planning and Real-time Vision**", International conference on *Field and Service Robotics* (FSR97), Canberra, Australia, 1997.

[Kanade 83] Kanade, T., and Kender J. R., "**Mapping Image Properties into Shape Constraints: Skew Symmetry, Affine-Transformable Patterns, and the Shape-from-Texture Paradigm**", *Human and Machine Vision*, Beck, Hope, and Rosenfeld, Eds., Academic Press, 1983.

[Kapur 85] Kapur, D., et al., "**Reasoning about Three Dimensional Space**", *Proceedings of the International Conference on Robotics and Automation*, St. Louis, MO, 1985, pp. 405-410.

[Kube 94] Kube, C. Ronald, Zhang, Hong, "**Collective Robotics: From Social Insects to Robots**", *Adaptive Behaviour*, Vol. 2, No. 2, 189-218. 1994.

[Kuniyoshi 94] Kuniyoshi, Yasuo, Rougeaux, Sebastien, Ishii, Makoto, Kita, Nobuyuki, Sakane, Shigeyuki and Kakikura, Masayoshi, "**Cooperation by Observation [The framework and basic task patterns]**", *Proceedings of the 1994 IEEE International Conference on Robotics and Automation*, San Diego, California, May 8-13, 1994.

[Mataric 95] Mataric, Maja J., "**Issues and Approaches in the Design of Collective Autonomous Agents**", to appear in "*Moving the Frontiers Between Robotics and Biology*", special issue of Robotics and Autonomous Systems, Vol. 16, Nos. 2-4, December 1995.

[Maybeck 79] Maybeck, P., "**Stochastic Models, Estimation and Control**", *Vol 1, Academic Press*, 1979.

[McLauchlan 94] McLauchlan, P.F., Reid, I.D., Murray, D.W., "**Recursive Affine Structure and Motion from Image Sequences**", *Lecture Notes in Computer Science Vol.800, Proceedings of ECCV'94*, pp 217-224, ISBN 3-540-57956-7, Springer Verlag Berlin Heidelberg 1994.

[Noreilis 92] Noreilis, Fabrice R., "**An Architecture for Cooperative and Autonomous mobile Robots**", *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, Nice, France - May 1992.

[Parker 94] Parker, Lynne E., "**An Experiment in Mobile Robotic Cooperation**", *Proceedings of the ASCE Specialty Conference on Robotics for Challenging Environments*, February, 1994: 131-139.

[Parker 95] Parker, Lynne E., "**The Effect of Action Recognition and Robot Awareness in Cooperative Robotic Teams**", *IEEE 0-8186-7108-4/95*, 1995.

[Steels 90] Steels, Luc, "**Cooperation between Distributed Agents through Self-Organisation**", *VUB AI Lab*, Pleinlaan 2, 1050 Brussels Belgium, IEEE *IROS 90*.

[Yuta 90] Yuta, Shin'ichi and Premvuti, Suparerk, "**Consideration on the Cooperation of Multiple Autonomous Mobile Robots**", *IEEE International Workshop on Intelligent Robots and Systems* (IROS '90).

[Yuta 91] S. Yuta, S. Suzuki and S. Iida, "**Implementation of a small size experimental self-contained autonomous robot - sensors, vehicle control, and description of sensor based behavior**", *Proc. Experimental Robotics*, Tolouse, France, LAAS/CNRS (1991).