A parallel implementation of a structure-from-motion algorithm

Han Wang[1] *Chris Bowman*[2] *Mike Brady*[1] *and Chris Harris*[3]

[1] Oxford University, Robotics Research Group, Oxford, OX1 3PJ, UK
[2] DSIR Industrial Development, 24 Balfour Road, Auckland, NZ
[3] Roke Manor Research Ltd, Roke Manor, Romsey, SO51 0ZN, UK

Abstract. This paper describes the implementation of a 3D vision algorithm, Droid, on the Oxford parallel vision architecture, PARADOX, and the results of experiments to gauge the algorithm's effectiveness in providing navigation data for an autonomous guided vehicle. The algorithm reconstructs 3D structure by analysing image sequences obtained from a moving camera. In this application, the architecture delivers a performance of greater than 1 frame per second – 17 times the performance of a Sun-4 alone.

1 Introduction

PARADOX [5] is a hybrid parallel architecture which has been commissioned at Oxford in order to improve the execution speed of vision algorithms and to facilitate their investigation in time-critical applications such as autonomous vehicle guidance. Droid[3] is a *structure-from-motion* vision algorithm which estimates 3-Dimensional scene structure from an analysis of passive image sequences taken from a moving camera. The motion of the camera (*ego-motion*) is unconstrainted, and so is the structure of the viewed scene. Until recently, because of the large amount of computation required, Droid has been applied off-line using prerecorded image sequences, thus making real-time evaluation of performance difficult.

Droid functions by detecting and tracking discrete image features through the image sequence, and determining from their image-plane trajectories both their 3D locations and the 3D motion of the camera. The extracted image features are assumed to be the projection of objective 3D features. Successive observations of an image feature are combined by use of a Kalman filter to provide optimum 3D positional accuracy.

The image features originally used by Droid are determined from the image, I, by forming at each pixel location the 2×2 matrix, $A = w * [(\nabla I)(\nabla I)^T]$, where w is a Gaussian smoothing mask. Feature points are placed at maxima of the response function R [3], $R = det(A) - k(trace(A))^2$, where k is a weighting constant. Often, features are located near image corners, and so the operator tends to be referred to as a corner finder. In fact, it also responds to local textural variations in the grey-level surface where there are no extracted edges. Such features arise naturally in unstructured environments such as natural scenes. Manipulation and matching of corners are quite straightforward and relatively accurate geometric representation of the viewed scene can be achieved. In the current implementation, the depth map is constructed from tracked 3D points using a local interpolation scheme based on Delaunay triangulation [2].

Droid runs in two stages: the first stage is the booting stage, called boot mode, in which Droid uses the first two images to start the matching process; the second stage is the run stage called run mode.

In the boot mode, points in the two 2D images are matched using epipolar constraints. The matched points provide disparity information which is then used for estimation of ego-motion and 3D instantiation. Ego-motion is described as a 6-vector (3 in translation and 3 in rotation).

The run mode of Droid includes a 3D-2D match which associates the 3D points with the newly detected 2D points, an updated ego-motion estimation and a 2D-2D match, between residual points in the feature points list and unmatched points from the previous frame, to identify new 3D features. Also, 3D points which have been unmatched over a period are retired.

2 PARADOX Architecture

PARADOX is a hybrid architecture, designed and configured especially for vision/image processing algorithms. It consists of three major functional parts: a Datacube pipelined system, a transputer network and a Sun4 workstation. The architecture of PARADOX, as applied to Droid, is shown in Figure 1.

The Datacube family contains more than 20 types of VME-based pipelined processing and input/output modules which can perform a wide range of image processing operations at video rates. Image data is passed between modules via a Datacube bus-standard known as the MaxBus. System control is by means of the VME bus from the host Sun workstation. The Datacube can be used for image digitisation, storage and display and also for a wide range of video frame rate pixel-based processing operations.

The transputer network consists of a fully populated Transtech MCP 1000 board. This contains 32 T800 transputers – each with one Mbyte of RAM – and both hardwired and programmable switching devices to allow network topology to be altered. A wide range of network topologies can be implemented including parallel one dimensional arrays [6], a 2D array or a ring structure. This board delivers a peak performance of 320 MIPS. The connection between the Datacube and the transputer network is by way of an interface board designed by the British Aerospace Sowerby Research Centre [4].

In the parallel implementation of Droid [5], the Datacube is used to digitise, store and display the image sequences and graphics overlays; the corner detection is carried out by the transputer array and the 3D-isation is computed on the Sun workstation.

3 Performance Evaluation

Figure 2 shows an image from a sequence of frames with a superimposed Cartesian grid plot of the interpreted 3D surface by Droid. The driveable region can be clearly identified. An algorithm has been developed by D. Charnley [1] to extract the drivable region by computing the surface normal of each grid.

The above demonstrates qualitatively the performance of Droid in live situations, but not quantitatively. A series of experiments has been conducted at Oxford and at Roke Manor to measure the performance of Droid in both live and static environments. The intention has been to demonstrate the competence of dynamic vision in a real environment.

The performance obtained from PARADOX for parallel Droid was 0.87 seconds per frame which is 17 times faster than a pure Sun-4 implementation. The overall performance is limited primarily by the parallel execution of the 3D-isation and corner detection algorithms which have comparable execution times. The Datacube control and visual display functions contribute negligible fraction of execution time.

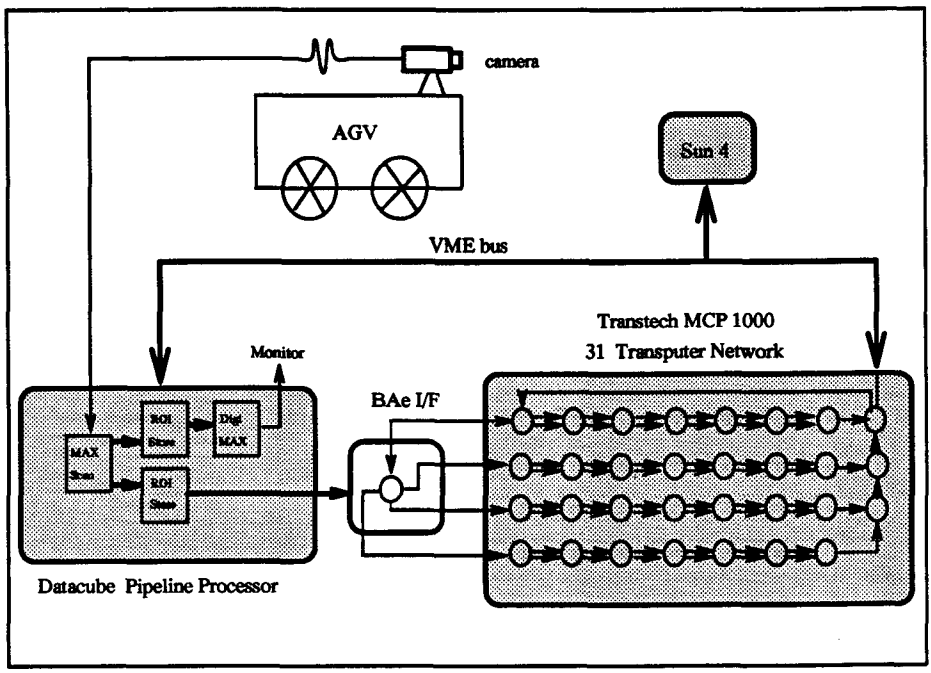

Fig. 1. *Machine architecture of PARADOX (Droid incarnation)*

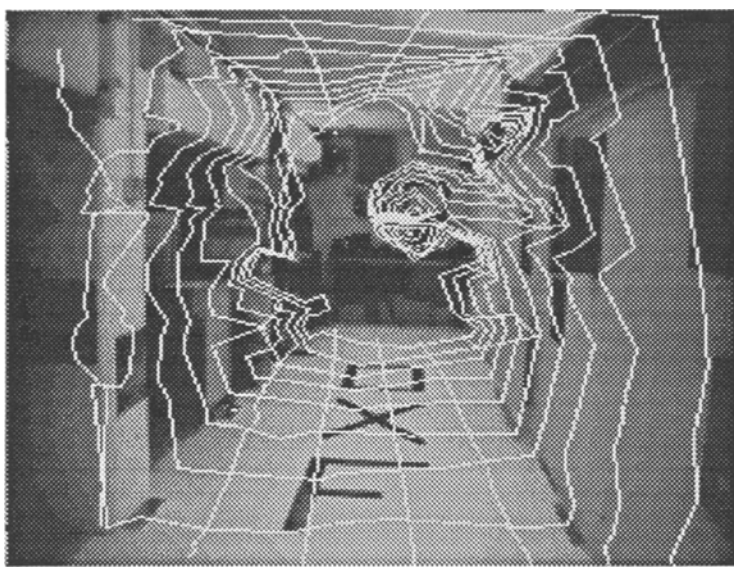

Fig. 2. *Droid reconstructed 3D surface, a driveable region can be clearly identified*

The laser scanner on the vehicle can determine the AGV's location (2D position and orientation) by detecting fixed bar-coded navigation beacons. This allows comparison between the "true" AGV trajectory and that predicted by Droid. The following results were obtained from an experiment where the AGV was programmed to run in a straight line with varying speeds.

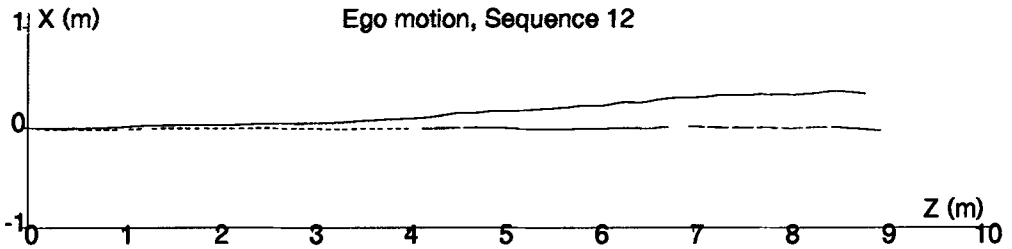

Fig. 3. *Plane view of AGV trajectory. Solid line–Droid predicted motion; Dashed line–laser scanner readings*

Figure 3 depicts a plane view of the AGV's trajectories: the solid line represents the AGV trajectory reported by Droid and the dashed line as reported by the laser scanner. In this particular run, the AGV has been programmed to move in a straight line at two different speeds. For the first part of the run it travels at about 8 cm/sec and for the second it travels at about 4 cm/sec. Droid reports the camera position (6 degrees of freedom – 3 translation and 3 rotation) from the starting point of the vehicle, which has coordinates $(x_0, z_0) = (0, 0)$ and the laser reported trajectory is re-aligned accordingly. It can be seen from Figure 3 that the alignment between the laser scanner readouts and the Droid prediction is very close.

During the run, the vehicle has been stopped twice manually to test this system's tolerance under different situations. Figure 4 shows the speed of the AGV as determined by Droid (solid line) and by the on-board laser scanner(dashed line). The speed plots in figure 4 agree closely apart from the moment when the vehicle alters its speed where Droid consistently overshoots. This can be improved using non-critical dumping.

4 Conclusion and future work

Droid constructs an explicit three-dimensional representation from feature points extracted from a sequence of images taken by a moving camera. This paper has described the algorithm, the PARADOX parallel vision architecture and the implementation of Droid on PARADOX. Experiments have demonstrated the competence of Droid and the performance of PARADOX in dealing with real world problems. The results show that this system is capable of identifying basic surface structure and can be used to supply purely passive guidance information for autonomous vehicles where other sensory mechanism finding it hard or impossible. Recently, an improved corner detection algorithm has

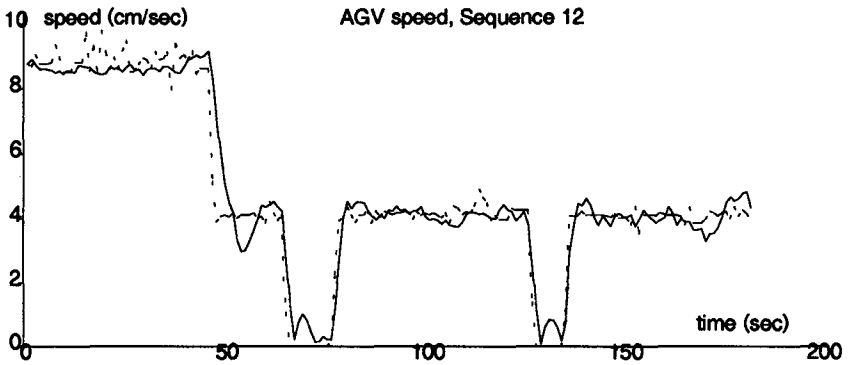

Fig. 4. *Comparison of AGV speed. Solid line–speed reported using Droid; Dashed line–speed reported using laser scanner*

been developed and is under test at Oxford [7]. This uses second order directional derivatives with the direction tangential to an edge. This algorithm has improved accuracy of corner localisation and reduced computational complexity. Consequently, it allows faster execution speed (14 frames per second) than the original Droid corner detection algorithm. This, together with parallelisation of the 3D-isation algorithms, will offer further improvements to overall execution speed. Future work will include (1) the incorporation of the new fast corner detection algorithm into Droid, (2) the use of odometery information taken from the AGV to provide Droid with more accurate motion estimations, and (3) to eventually close the control loop of the AGV–that is to control AGV by utilising the information provided by Droid.

References

1. D Charnley and R Blisset. Surface reconstruction from outdoor image sequences. *Image and Vision Computing*, 7(1):10–16, 1989.
2. L. De Floriani. Surface representation based on triangular grids. *The Visual Computer*, 3:27–50, 1987.
3. C G Harris and J M Pike. 3D positional integration from image sequences. In *Proc. 3rd Alvey Vision Conference*, Cambridge, Sept. 1987.
4. J.A. Sheen. A parallel architecture for machine vision. In *Colloquium on Practical applications of signal processing.* Institution of the Electrical Engineers, 1988. Digest no: 1988/111.
5. H Wang and C C Bowman. The Oxford distributed machine for 3D vision system. In *IEE colloquium on Parallel Architectures for Image Processing Applications*, pages 1/2–5/2, London, April 1991.
6. H Wang, P M Dew, and J A Webb. Implementation of Apply on a transputer array. *CONCURRENCY: Practice and Experience*, 3(1):43–54, February 1991.
7. Han Wang and Mike Brady. Corner detection for 3D vision using array processors. In *BARNAIMAGE 91*, Barcelona, Sept. 1991. Springer-Verlag.