Dynamo: real-time experiments with multiple mobile robots

R. Barman, S. Kingdon, J. J. Little, A. K. Mackworth,
D. K. Pai, M. Sahota, H. Wilkinson, Y. Zhang *
Computer Science Department
University of British Columbia
Vancouver, B.C., Canada
email: dynamo@cs.ubc.ca

Abstract

In the Dynamo project at UBC, we have developed an extensible facility for experimental mobile robotics. It currently consists of nine radio-controlled mobile robots, two CCD colour video cameras, a video transmitter and tuner, radio controllers, Datacube image processing hardware, and a network of transputers. Software for tracking, control, and simulation has also been developed. Ongoing projects include soccer playing, and constraint programming for robots. We describe the facility and outline the projects.

1 Introduction

Dynamo is an umbrella project for research in real-time control of mobile robots. We have developed an extensible experimental facility for mobile robotics with a large number of mobile robots and real-time vision. This includes the Dynamite testbed consisting of radio controlled cars whose absolute position is sensed using off-board vision and which receive commands from a remote computer. The facility is used by several projects for experimental research in mobile robots.

For example, we are currently attempting to have teams of robots engage in cooperative and competitive behavior such as playing a game of soccer or hockey. This task is simple to state but challenging to perform with real robots, and it requires solutions that integrate sensing, planning, and acting. It is necessary to sense the locations of the robots, ball, and goal in real time, to plan strategies for scoring and defending, and to control the actions of the robots in the presence of uncertainty and non-holonomic constraints. Figure 1 depicts two mobile robots playing soccer in our lab.

*Alphabetical order.

Multiple robots have attracted considerable interest recently recently [Kub92, Nor92, Mat92]. Several experiments have been reported in this area, but they have been restricted to simple tasks and cooperative behaviour. Competition between an agent and a hostile environment has been addressed in [AC87]; however, this work involved a software agent operating in a simulation, and not in the real world.

In section 2 we describe the experimental hardware which allows the control of several mobile robots with real-time vision. Section 3 describes the software interface for controlling the robots from high level programs running on a network of transputers. A simulator for program development and debugging is also provided. Section 4 outlines some projects using the experimental facility.

2 Hardware Environment

Figure 2 depicts the overall hardware architecture of the Dynamo facility.

The mobile robot bases are commercially available radio-controlled vehicles with proportional speed and steering control. These include two Tamiya 1/10 scale "monster" trucks (Figure 1), and seven 1/24 scale race cars (Figure 3). The low unit cost of each chassis permits experiments with a large number of robots. Each robot is controlled using an R/F transmitter connected to the transputer network. There are two CCD colour video cameras connected to the Datacube hardware: (1) an off-board camera, a fixed ceiling-mounted camera with an overhead view of the mobile robot pen and (2) an on board miniature camera, a pannable camera with a broadcast video link.

The Vision Engine [LBKL91] (see Fig 2) consists of a Datacube MaxVideo-200 image processor with a Digicolor colour image digitizer, and a MIMD multicomputer consisting of 20 T800 transputers operating at 25 MHz with programmable interconnections

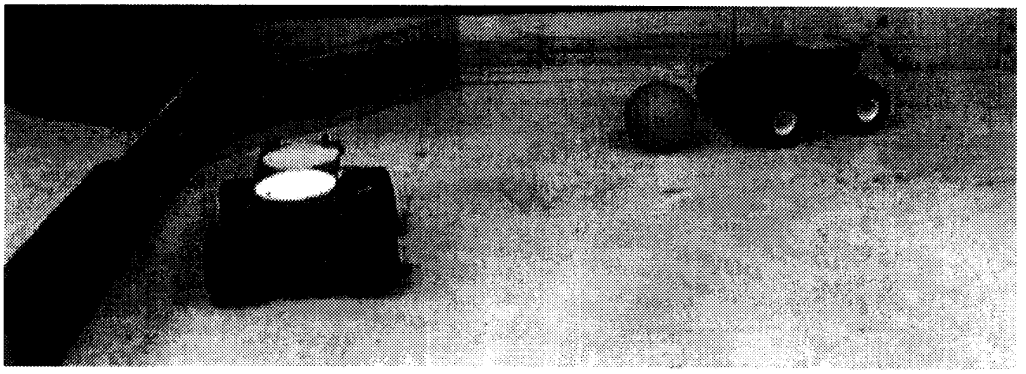

Figure 1. Two Dynamo mobile robots playing soccer

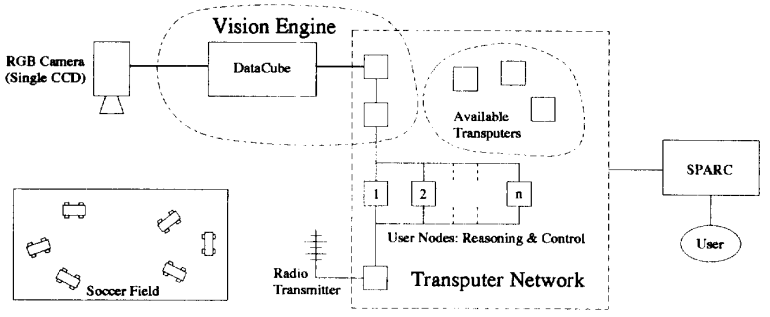

Figure 2. Dynamo Architecture

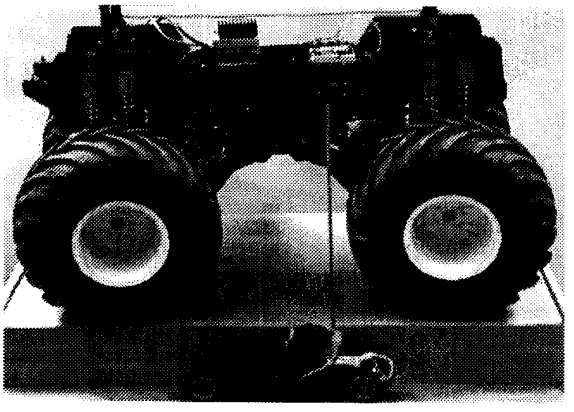

Figure 3. Dynamo robots: Monster Truck "Heraclitus" and Dynamite "Zeno"

through a crossbar switch. These subsystems are connected by a Maxtran board, a bidirectional interface operating at video rates. The Maxtran board maps data from the video bus of the Datacube system into video RAM attached to a transputer, where image data can be partitioned and sent to other processors in the system. The effective data rate on the video bus is 7.0MB/s which matches well the combined data rate of 6.7MB/s across the four unidirectional links exiting the transputer on the Maxtran. The communication is bidirectional so that data can be returned to the Datacube system for further processing or display. Figure 2 depicts the organization of our system. The entire system resides on the VME bus connected to a Sun SparcStation 2 host.

In addition to the Dynamo robots, the system controls a CRS-460 six degree-of-freedom robot arm and other actuators such as the UBC eye-head from a transputer. As well, the transputer system has direct image digitizing and image display.

3 Software Environment

We have developed software to track the mobile robots at servo rates using vision, and to control them from programs running on the transputers. The control software lets users write high-level programs for robot motion in world coordinates. The tracking software locates objects at video rates using colour markers. The centroid of each distinct visible colour class is computed and transformed to world coordinates at 60Hz, enabling real time control.

3.1 Tracking

The Vision Engine is able to track the world space position of the centroids of coloured targets placed on the robots at 60Hz using the off-board camera.

The colour video camera is placed in a fixed position so that the entire arena/workspace is visible. The video is fed into the Datacube hardware which is programmed to preprocess the image to simplify and speed up the centroid calculations. The incoming colour pixels are converted from an RGB colour space into an HSV colour space and then classified either as background or as belonging to some colour class. This stream of classified pixels is then run-length encoded and passed onto the transputers for further processing.

The Maxtran transputer examines the incoming processed image and finds the centroid, in screen coordinates, of each disjoint, connected region, assuming convexity. These coordinates are then passed onto a second transputer that corrects for radial lens distortion and finds the corresponding point in world space which satisfies both the perspective projection for the camera's view position and the constraint that all points must lie in a known plane (parallel to the floor).

Once the world space position of each region has been found, it is necessary to map these onto targets. In the current implementation there will be only a single target of each colour class visible, and the region with the most pixels for each class is assumed to be the target. Then, since each robot has two targets on it, the position and orientation of the robot can be found directly from the position of its targets.

This system has been found to be very robust in practice. By painting the targets using fluorescent paint, and then only accepting highly saturated colours of the appropriate hue, the pixel classification can be made with almost perfect accuracy. A further advantage of picking fluorescent colours is that it is less sensitive to changes in the colour of the lighting.

The system is quite fast, for tracking with video cameras, and has low latency. The Datacube preprocessing only introduces a few milliseconds of delay relative to the camera, and the Maxtran transputer typically needs only 7-10ms of processing time to find the region's centroids (for a normal image this processing can be done while the image is being transferred to the Maxtran). Hence, since the remaining calculations are fairly trivial, the tracking is able to run at 60Hz and finds all objects within a couple of milliseconds of the time that the camera has finished transmitting the image.

3.2 Control

We currently have a two-level control interface to the robots. The low level programming model for the control of each car consists of a single function call that sets the "throttle" and desired steering angle of the car. The next level control interface accepts desired forward speed v and turning radius ρ. The actual speed and turning radius are estimated and a PI controller servos the robot to maintain the goals.

All control requests are handled by a single transputer which has a separate R/C transmitter for each robot. The transputer translates throttle and steering commands received via one of its links into control packets which are sent via a pulse width modulated AM signal to the R/C receivers mounted on the robots. Each receiver decodes and demultiplexes the control packets and uses it to drive the robot's servos.

In AM R/C equipment, the control packet is eight

or less pulses varying in duration between 0.5ms and 1.5ms and repeating every 20ms (i.e. at 50Hz). The transmitters each have a single control wire which is connected to a digital I/O port on the transputer that can be toggled under software control. The transputer is more than fast enough to toggle the state of its ports under software control and to asynchronously receive and format new control requests.

3.3 Simulation

We have implemented a simulator for developing and testing mobile robot programs. The programs can be executed without change on the simulator and the resulting motion can be viewed with 3D interactive computer graphics in real time on a workstation. This facilitates rapid prototyping and debugging of robot programs.

The simulator runs under the stand-alone version of the Trollius multicomputer operating system [Tro91]. The robots are modeled using empirical data that was collected during test runs with the actual robots. Also provided is the emulation of the hardware interface and which allows a controller process to communicate, through Trollius, with the simulator process.

The Datacube and transputers are single-user, special-purpose hardware, with non-trivial set-up time. User conflicts over their usage often arise, which makes developing algorithms using the hardware inconvenient and time-consuming. Experience has shown that developing the algorithm prototype first with the simulator and later fine-tuning it with the hardware is much faster than using the hardware alone.

4 Projects

The Dynamo project supports experimental investigation for several research projects. We describe these below.

4.1 Constraint Nets

The Constraint Net (CN) approach of Zhang and Mackworth is a model for robotic systems software implemented as modules with I/O ports [ZM92b]. A module performs a transduction from its input traces to its output traces, subject to the principle of causality: an output value at any time can depend only on the input values before, or at, that time. The language has a formal semantics based on the least fixpoint of sets of equations [ZM92a]. This approach allows one to specify formally, and verify, models of embedded control systems. Our goal is to develop it as a practical tool for building real, complex, sensor-based robots. It can be seen as a development of Brooks' subsumption architecture [Bro88, Mac93] that enhances its modular advantages while avoiding the limitations of the augmented finite state machine approach.

A robot situated in an environment is modeled as three machines: the robot plant, the robot control and the environment. Each is modeled separately as a dynamic system by specifying a CN with identified input and output ports. The robot is modeled as a CN consisting of a coupling of its plant CN and its control CN by identifying corresponding input and output ports. Similarly the robot CN is coupled to the environment CN to form a closed robot-environment CN.

The CN model can be realized as an on-line distributed programming language with a formal algebraic denotational semantics. Furthermore, we have developed a specification language, a real-time temporal logic, that allows the designer to specify and prove properties of the situated robot by proving them of the robot-environment CN. So far, we have been able to specify, design, verify and implement systems for a robot that can track other robots [ZM92b], a robot that can escape from mazes and a two-handed robot that assembles objects [ZM92c], an elevator system [ZM93] and a car-like robot that can plan and execute paths under non-holonomic constraints. Experimental verification of CN approach using the Dynamo facility is in progress.

4.2 Least Constraint

Pai has proposed an approach called Least Constraint (LC) for programming high degree of freedom robots using time- and state-dependent assertions [Pai91]. These assertions are defined using inequality constraints which describe the set of allowed states as a function of time. The constraints are solved at run time to produce a motion satisfying them.

Since complex mechanical systems have large state spaces, it is not convenient or natural to express all of the constraints in a single space. For convenience of expression, users define derived variables in terms of the basic (e.g., state) variables — an example of this is the definition of task and end-effector coordinates for robot manipulators. LC generalizes such constructions to allow the creation of arbitrary, user definable quantities which are natural to the tasks and the constraints being expressed. One can isolate small groups of variables into domains $\mathcal{D}^i$ on which to focus. $\mathcal{D}^i$ are differentiable manifolds with an atlas

of charts.

A *motion specification* in LC consists of a system of time-varying inequality constraints $P_\alpha, \alpha \in A$, on the domains $\mathcal{D}^i$: here each constraint P_α is expressed by

$$P_\alpha := f_\alpha(\mathbf{x}, \dot{\mathbf{x}}, t) \leq 0,$$

where $f_\alpha : \mathcal{D}^i \times T\mathcal{D}^i \times \mathbb{R} \rightarrow \mathbb{R}$, is a smooth map, and $\mathbf{x}(t)$ denotes a time-dependent trajectory in $\mathcal{D}^i$. Such P_α and their conjunctions

$$P := \bigwedge_\alpha P_\alpha$$

are executable LC motion programs. The meaning of the constraint function is that the system is controlled to make the specified expressions $P_\alpha := f_\alpha(\mathbf{x}, \dot{\mathbf{x}}, t) \leq 0$ true at all times t.

The constraint satisfaction is currently performed at run time by discretizing the derivatives using an implicit Euler method, and then using a quadratic penalty conjugate direction method to solve the discretized problem. Fast derivative computation is performed using automatic differentiation [PS93].

We are currently experimenting with the LC approach to program the Dynamo mobile robots. While mobile robots are relatively low degree of freedom machines, a collection of mobile robots may be viewed as a single high degree of freedom robot. In addition, it is difficult to specify the motion of a wheeled robot due to the presence of non-holonomic constraints. Let the configuration space of a mobile robot be $\mathcal{C}$ with coordinates x, y, θ ($\mathcal{C}$ is $E(2)$ the space of Euclidean motions in the plane). It is well known [Lat91] that the condition that the wheels of the robot roll without slipping impose a constraint

$$\dot{x} \sin\theta - \dot{y} \cos\theta = 0$$

that is non-integrable. Therefore the constraint does not lower the dimension of configuration space, but on the other hand, the user cannot specify an arbitrary trajectory $(x(t), y(t), \theta(t))$ to track. By specifying the desired motion weakly in terms of the constraints on the required motion, the constraints can be solved at run-time to produce a motion that satisfies the constraint.

4.3 Intelligent Action Selection

Sahota has proposed a two layer architecture for a robot controller. The lower layer is composed of a set of task-following modules, each of which can interact with the environment to perform a well defined task or activity. This follows from similar work by [Fir92] . The deliberative layer of our controller is composed of behaviour producing modules. The behaviour based approach is common to much of the work on the situated agents. However, we feel that the current mechanisms for arbitrating among behaviours are inadequate.

Our method for intelligent action selection is based on inter-behaviour bidding. This approach is more general than other approaches theory of action selection is that each behaviour is best able to identify how applicable it is in a given situation. Each behaviour independently evaluates the world and reports a utility estimate or bid to the other behaviours. The behaviour with the highest bid assumes control since it has the greatest utility. The allowable bid ranges are set by the designer, just as the hierarchy of behaviours is set in the Subsumption Architecture. One advantage of a this scheme is that the bid ranges can be set at run-time or even changed on the fly in a system which learns. To clarify, there is little relation between our approach and other systems like Contract Nets [Smi80, Nor92] where there is negotiation between behaviours.

This theory of action selection can be extended to multiple robots. In this case, each robot would broadcast its intended actions and a bid which estimates the appropriateness of that action. Robots whose actions are in conflict will reevaluate the situation including the bid information of other robots. Robots with lower bids than other robots will lower their internal bid for that action which will result in the selection of some other action.

The effectiveness of the lower layer of our architecture has been demonstrated using the Dynamite testbed. Controller functionality includes motion planning, ball shooting, and playing goal. The arbitration scheme in the higher layer has not yet been implemented. We have, however, had two robots compete in a one-on-one game of soccer. The controller for each robot alternates between striker and goalie modes. This does not result in particularly intelligent behaviour, but it does demonstrate the functionality of the lower layer of control and the usefulness of the Dynamite testbed. We have a video which shows the testbed and documents the two robots playing soccer. We expect to have two teams with three robots on each side playing soccer in the near future.

5 Conclusions

We have designed an experimental facility to support mobile robotics projects in several areas, including collaboration and competition among multi-

ple agents; control of high degree of freedom systems; navigation; real-time vision; planning and acting in dynamic and uncertain environments. Designing the facility around low cost radio controlled robots and high performance real-time vision has brought real benefits. Results to date include a soccer-playing mobile robot that can score goals and frameworks for programming mobile robots with constraints.

Acknowledgments

This work is supported in part by NSERC and the Institute for Robotics and Intelligent Systems. Alan Mackworth is the Shell Canada Fellow of the Canadian Institute for Advanced Research.

References

[AC87] Philip Agre and David Chapman. Pengi: An implementation of a theory of activity. In *Proceedings of AAAI-87*, 1987.

[Bro88] R. A. Brooks. *A robot that walks: emergent behaviors from a carefully evolved network.* Massachusetts Institute of Technology, Cambridge, MA, 1988.

[Fir92] R. James Firby. Building symbolic primitives with continuous control routines. In *First International Conference on Artificial Intelligence Planning Systems*, 1992.

[Kub92] Claus Ronald Kube. Collective robotic intelligence: A control theory for robot populations. Master's thesis, University of Alberta, 1992.

[Lat91] J. C. Latombe. *Robot Motion Planning.* Kluwer, 1991.

[LBKL91] James J. Little, Rod Barman, Stewart Kingdon, and Jiping Lu. Computational architectures for responsive vision: the vision engine. In *Proceedings of CAMP-91, Computer Architectures for Machine Perception*, pages 233–240, December 1991.

[Mac93] A. K. Mackworth. On seeing robots. In A. Basu and X. Li, editor, *Computer Vision: Systems, Theory, and Applications.* World Scientific Press, Singapore, 1993. (in press).

[Mat92] Maja Mataric. Minimizing complexity in controlling a mobile robot population. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, 1992.

[Nor92] Fabrice Noreils. An architecture for cooperative and autonomous mobile robots. In *Proceedings of the 1992 IEEE International Conference on Robotics and Automation*, 1992.

[Pai91] Dinesh K. Pai. Least constraint: A framework for the control of complex mechanical systems. In *Proceedings of the American Control Conference*, pages 1615 – 1621, 1991.

[PS93] D. K. Pai and T. H. S. Ser. Simultaneous computation of robot kinematics and differential kinematics with automatic differentiation. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems '93*, 1993.

[Smi80] G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computing*, 29(12), 1980.

[Tro91] The trollius user's reference manual. Research Computing, The Ohio State University, March 1991.

[ZM92a] Y. Zhang and A. K. Mackworth. Constraint nets: A semantic model for real-time embedded systems. Technical Report TR 92-10, UBC, Vancouver, B.C., May 1992.

[ZM92b] Y. Zhang and A. K. Mackworth. Modeling behavioral dynamics in discrete robotic systems with logical concurrent objects. In *Robotics and Flexible Manufacturing Systems*, pages 187–196. Elsevier Science Publishers B.V., 1992.

[ZM92c] Y. Zhang and A. K. Mackworth. Will the robot do the right thing? Technical Report TR 92-31, UBC, Vancouver, B.C., November 1992.

[ZM93] Y. Zhang and A. K. Mackworth. Design and analysis of embedded real-time systems: An elevator case study. Technical Report TR 93-4, UBC, Vancouver, B.C., February 1993.