Reactive low level control of the ARK

Matt Robinson and Michael Jenkin
Department of Computer Science, York University,
4700 Keele St., North York, Ontario, Canada
email: (robinson,jenkin)@cs.yorku.ca

Abstract

This paper describes the design and implementation of a robot guidance system which integrates a classical occupancy grid based global path planner to construct a global plan and a low-level subsumption based architecture to safely execute it. The global path planner generates a correct path from the robots current state to a goal state according to a static map of the world, and the low-level architecture is given the task of avoiding any unexpected or unmodelled obstacles. By dividing the task of driving the robot to a mapped goal into a classical AI task and a reactive subsumption one, the ARK robot takes advantage of the best of both technologies. Results obtained using a RWI B12 mobile base, an onboard reactive control system and the ARK global path planner are shown.

1 Introduction

ARK (Autonomous Robot for a Known environment) is a visually guided mobile robotics project undertaken as a collaboration between Ontario Hydro, Atomic Energy Canada Limited (AECL), The National Research Council of Canada (NRC), and two universities; York University and the University of Toronto. The goal of this project is to build an autonomous vehicle capable of performing inspection and monitoring tasks in a typical industrial environment. The project will eventually construct two different robots; a tethered robot, called ARK-1, which will be used as a proof of concept vehicle, and a second generation vehicle, called ARK-2, which will utilize the best of the results of ARK-1 and which will operate in real time with the majority of its computation performed onboard. Both robots use visual data obtained through active vision processes as the primary source of sensing for the robot, but both robots will also utilize non-visual sensors such as infrared, sonar and laser range finders. A general description of the ARK-1 robot, its design, implementation and goals can be found in [10].

The design of the ARK-1 robot (which will be referred to as ARK) has been driven by the final application domain of ARK-2. The ARK-2 robot will operate in a complex industrial environment, similar to the types of environments found in industrial bays, power plants, and other large complex manufacturing centers. These environments are radically different from the lab environments typically used in robotic research. In industrial environments the lighting varies dramatically from location to location and even by time of day. There are a number of permanent structures in the environment (external walls, pillars, etc.), but there are also a number of large semi-permanent structures (manufacturing cites, mockups, etc.), and even temporary structures. In addition the local environment is considerably more complex than those found in most robotic research labs. The floor contains cable trays, drainage ditches, and the like. Objects protrude at arbitrary heights and from arbitrary orientations. Walls, in the classic sense, may only be found at the environments external boundaries.

It is not practical in the ARK-2 environment to perform modifications of the environment such as adding bar codes to the walls, magnetic strips beneath the floor, or active radio beacons in order to simplify the problem of navigation, and as an industrial environment lacks the wall structure of an office or lab space, navigation by necessity has to rely primarily on landmark detection, identification and tracking (see [2,8,9,7]).

In order for the ARK robot to be able to carry out its required tasks it is essential that the robot maintain its position with respect to a global map. Thus a great deal of the computational effort of the robot is dedicated towards establishing and maintaining this correspondence. The robot utilizes known premapped landmarks in its environment coupled with odometry information to continually estimate its position with respect to the

global map. These landmarks are identified and localized using a combined laser rangefinder/colour video camera mounted on a computer controlled pan and tilt unit[4]. As maintaining the correspondence between the robots position and the world map is critical, the ARK path planner plans paths which minimize the maximum expected uncertainty of the robots position while moving[5].

The ARK robot must be able to complete a given task in a finite amount of time. When presented with a task such as "go and point the ARK video sensor at dial 3", the robot must either (i) compute a path that allows dial 3 to be acquired with the video sensor and then execute that plan in a finite length of time or (ii) report that it cannot perform the required task. This correctness requirement precludes robotic solutions which are purely reactive and rely on probabilistic solutions to the robots missions. Solutions such as "wander randomly pointing the camera everywhere and eventually dial 3 will be sensed" are not acceptable. Conversely, the ARK robot must be safe. The robot must be capable of dealing with unmodelled obstacles in the environment. This includes obstacles that may be passive but unmodelled, or moving obstacles such as people. These are tasks for which purely reactive systems have had some success but which are not well handled by "Good Old Fashioned AI and Robotics" (GOFAIR) approaches which assume a single active agent (the robot) and a static fully modelled world. (See [11] for a review of some classical GOFAIR approaches.) What is required in the ARK environment is a system which has the "correctness" of classical AI approaches but which also exhibits the flexibility or "safety" of reactive techniques.

2 The ARK planner

The ARK path planner is a hybrid of the classical GOFAIR approach and the subsumption approach. The planning task is broken down into two different phases. In the first phase a traditional path planner is used to obtain an optimal (under some metric) path from the robots current position to the goal. In a GOFAIR environment this plan could be executed directly as it assumed that the world is well modelled by the planner and the robot is the only active agent. As this is typically not the case, the plan is not executed directly, but is rather broken down into smaller subgoals (linear motions of the robot through the environment), and each of these subgoals are then fed sequentially to the second phase of the planner. The second phase is a reactive, subsumption-based architecture whose primary purpose is to execute plans obtained by the classical path planner while avoiding unexpected or unmodelled obstacles in the environment. The reactive planner is a safety critical real-time system modelled as a subsumption architecture.

While the reactive system is executing a subgoal, the GOFAIR path planner is free to take advantage of the inherent parallelism between the reactive planner and the classical planner. As the reactive system attempts to satisfy a subgoal, the GOFAIR system monitors its progress. Should the reactive system take an unreasonably long time to complete its subgoal or if the reactive system determines that it is unable to complete its task, the GOFAIR system identifies the reactive system as being incapabable of completing this subgoal, and chooses a different subgoal for the robot by replanning the global path. The GOFAIR planner assumes that the reactive system cannot accomplish this goal due to a permanent blockage on the straight line path that makes up the current subgoal. In the GOFAIR model, a permanent obstacle is placed directly in front of the current position of the robot, and a new classical path is determined that avoids the obstacle. Due to the grossly different time constants between the process of having the planner generate a path for the robot, and actually having the robot follow the path, it would be possible for the GOFAIR planner to continually postulate that the current plan will fail at some point just ahead of the current position of the robot, and to concurrently plan new alternative routes to the goal. (This is not currently implemented due to hardware limitations.)

2.1 The GOFAIR Path Planner

The GOFAIR ARK path planner is a classical occupancy grid based path planner similar to that proposed by Hwang and Ahuja[3]. The world is digitized into cells in (x, y, θ) and cell locations are assigned values indicating their distance to the nearest modelled structure in the environment. Given the starting pose of the robot and a goal pose, a breadth first search from the start pose to the goal is used to determine the optimal path (if one exists) from the start to the goal. At each step the robot can either move forward one cell or rotate one step in orientation. The ARK planner supports several different cost metrics for the breadth first search process. A shortest path, an occupancy grid minimization path, and a position uncertainty minimizing path are currently supported. The world is

```
landscape newworld;
  unit meter;
  width 22.500000 to 45.500000;
  height 12.000000 to 28.500000;
  elevation 0.000000 to 2.200000;
  begin
    name (43.541718,23.182617)
        "Elevator";
    name (33.433750,17.051640)
        "RoboticsLab";
    obstacle boundary2: wall;
      extent (39.849998, 25.200001) to
             (39.849998, 21.400000);
      elevation 0.000000 to 0.000000;
    end;
    obstacle boundary6: wall;
      extent (38.496956, 12.395000) to
             (38.496956, 12.065001);
      elevation 0.000000 to 0.000000;
    end;
end.
```

Figure 1: Sample environment language description

digitized in orientation as well as in position so that the motion costs associated with rotation changes can be modelled by the planner as can motions of robots having a non-circular cross section.

The ARK world is modelled as a collection of $2\frac{1}{2}$D primitives such as walls and pillars. A world modelling language which has some of the flavour of Pascal is used to specify the environmental layout (see Figure 1). Objects in the world have a two dimensional extent in (x, y) and have a constant range of elevations. This elevation information is not used by the path planner although it is used by the landmark acquisition process. The ARK world consists of obstacles, landmarks, and named places. Obstacles are of two types; walls, which are planar wall segments, and cylinders, which are circular wall segments. Named places are simply locations which have been given text labels, and landmarks are the visually identifiable landmarks used for robot localization.

Although the map can be created and edited by hand, a graphical editing tool is available to create the map in an interactive fashion. The map is represented within the path planner in an inverted fashion so as to be optimized for path planning operations. The current implementation makes no effort to optimize the internal representation of the world although refinements such as a quadtree representation could be used[12].

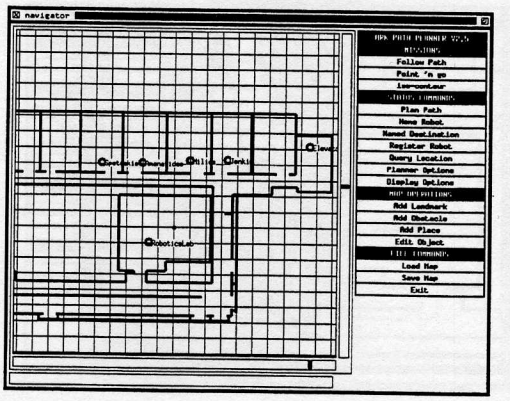

Figure 2: York research lab layout as modelled by the path planner. A 1 meter square grid has been overlayed over the map. The map shows various wall segments and some identified locations.

The ARK path planner has been used successfully in simulation mode, in driving a Cybermotion K2A platform through the University of Toronto AI Lab, and in driving a RWI B12 platform through the research lab space at York University. Detailed maps for both of these environments have been generated and part of the York research labs as modelled by the ARK path planner is shown in Figure 2. Given the radius of the robot, a particular grid size for digitizing the environment, and a map of the environment, the path planner can be used to plan GOFAIR paths throughout the environment, such as the one shown in Figure 3.

The paths obtained by the planner consist of a sequence of pose transitions of the robot such as $(x, y, \theta) \rightarrow (x + \delta x, y + \delta y, \theta + \delta\theta)$ where only one of δx, δy and $\delta\theta$ is non-zero. Although these individual pose transitions could be passed directly to the reactive phase of the planner, it was found to be more efficient to perform peephole optimization on the plan sequence before passing it to the reactive phase to be "executed". This peephole optimization involves concatenating pose transitions which when combined result in a single straight line motion of the robot, or in a single rotation of the robot.

The GOFAIR planner assumes the existence of a reactive subsystem which can be used to "execute" these optimize plans. Different reactive subsystems have been constructed; a simulator, a simple subsystem for the K2A Cybermotion platform which utilizes the K2A onboard sonar system and bumper system, and the RWI B12 unit described here. The GOFAIR planner communicates to the reactive planner through a very simple set of op-

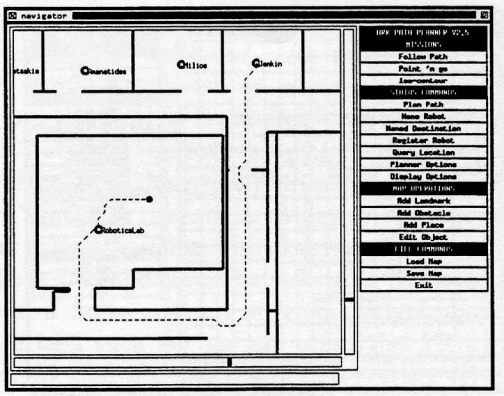

Figure 3: Path planner showing path. A path has been planned using the occupancy grid cost metric from a starting point in the Robotics lab to the office marked "Jenkin".

erations which assume that the reactive phase of the planner will operate autonomously and asynchronously attempting to achieve the current subgoal. The reactive planner supports the following messages relating to moving the robot base

SETPOS This operation sets the reactive control systems concept of where the robot currently is. As a side effect this message sets the goal pose to be the robots pose. This operation is performed when the robot is registered and homed with respect to the map.

GETPOS Read the reactive planners concept of where the robot currently is.

SETGOAL Provides the reactive planner with a new subgoal. The reactive planner continues to attempt to achieve this subgoal until it is presented with a new one.

GETSTAT Query the reactive planner about its current status. The planner responds with one of **MOVING** to indicate that it is currently moving towards its current goal, **GOAL** to indicate that it is currently at the goal, and **BLOCKED** to indicate that there is nowhere for the robot to go.

2.2 Reactive Control System

The low-level control of the robot is based upon the subsumption approach originally set forth in [1]. (See also [6].) The robot is guided by a set of behaviours which operate in parallel. Each behaviour maps sensory readings from the robot's environment into an external action of the robot.

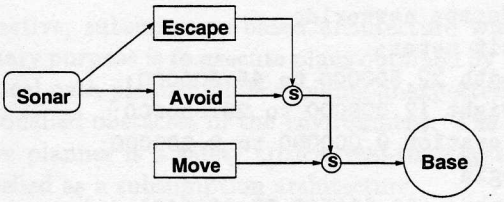

Figure 4: Basic subsumption architecture from [6].

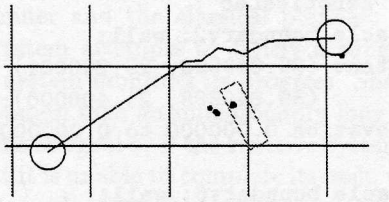

Figure 5: Robot moving through world. The robot starts at the upper right location and is given a goal position towards the lower left. The RWI robot senses the obstacle (shown by a grey outline) and navigates around it. Note that the robot is only given the relative position of the goal. It does not know of the presence of an obstacle. The small dots in the diagram show sonar hits which triggered the robot's Avoid behaviour.

Conflicting behaviours are arbitrated based on an absolute prioritization of behaviours. For example, a behaviour which detects and avoid obstacles would have precedence over a behaviour which simply translates the robot without regard to its environment.

A very simple subsumption architecture based on the one presented in [6] is used as the core of the ARK reactive control system (see Figure 4). There are three behaviours that control the robot. Two of these are dedicated to avoiding collisions and finding safer paths around obstacles. The first of these, and the most frequently used is the *avoid* behaviour. This behaviour watches for obstacles on the forward-facing sonar sensors. If an object appears, the *avoid* behaviour stops the robot, and turns it to a new direction so that the robot will not collide with the obstacle. The *escape* behaviour watches for an obstacle directly in front of the robot, in which case, it causes to robot to back-up and then turn to a new direction. The *escape* behaviour helps to get out of certain deadlocks that may occur with the *avoid* behaviour when the robot gets stuck in a corner. The *move* behaviour steers the robot towards a precomputed goal position.

The capabilities of the subsumption architecture are shown in Figure 5. The robot starts at the up-

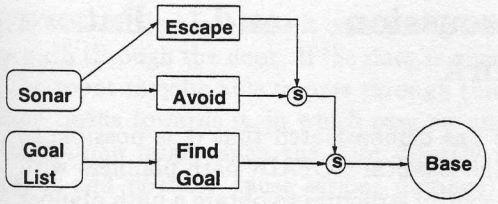

Figure 6: Modified reactive architecture.

per right hand location and is given as its goal position the lower left position. The robot base moves autonomously from the start position to the goal position. During its motion it senses the presence of obstacles and moves so as to avoid the sensed obstacle. Note that the RWI robot does not operate using a move and look paradigm but rather continually adjusts the speed of the vehicle.

The reactive control system is safe in that the avoid and escape behaviours will prevent the move behavior from driving the robot into an obstacle *provided that the obstacle is sensed by the sonar system*. The subsumption architecture is implemented on a single 386 laptop computer which uses a round robbin schedule to allocate computational resources to each of the behaviours. This delay, plus the delay associated with the onboard sonar system results in a maximum of roughly 1/15 of a second between the time of an event in the environment and the robot being able to react to it. This sets a maximum top speed for the robot in order for the *escape* behavior to be able to avoid having the robot strike a stationary object in its environment.

In order for the GOFAIR planner to use the subsumption planner to drive the robot safely from place to place, the subsumption architecture described in Figure 4 was modified so that rather than choosing a single goal, the reactive system is presented with a list of goals that it will attempt to achieve in sequence. This is shown in Figure 6 in which the *move* behavior has been modified not to use a single goal, but to rather use a list of goals as its input. The GOFAIR path planner generates a list of (x, y, θ) values and the reactive system attempts to satisfy them. The reactive path planner does not currently make use of the θ values.

3 Experiments and Implementation Details

Ideally, the GOFAIR and reactive control systems are combined in such a way that they are both housed on-board the robot or are tethered electronically. This is the approach taken with the Cybermotion K2A platform, which uses a wide spectrum radio modem to communicate between the onboard processors and the host computers. The Cybermotion platform lacks sufficient onboard computation and onboard sonar sensors in order to be able to implement a reactive control system as described here. For example, the Cybermotion sonar system cannot easily be changed to look behind the robot after looking in the forward direction as the robot powers down the sonar system in the backwards direction when looking forwards and vice versa. The RWI B12 base on the other hand is equipped with a sufficiently rich set of sensors so as to be able to implement a reactive control structure. The B12 is equipped with a RWI G96 sonar board which provides 12 Polaroid sonar sensors evenly spaced around the robot. The robot is roughly 38cm high and is about 27cm in diameter. A MS-DOS 80386 laptop computer mounted on top of the robot provides control signals to the base and sonar subsystems as well as providing a computational platform for the reactive control system. This is shown in Figure 7.

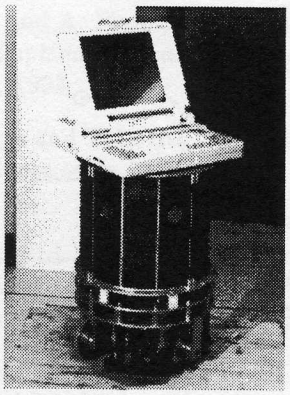

Figure 7: RWI B12 robot with onboard computer used for experimentation.

The 80386 machine mounted on the robot is not sufficiently powerful to provide both the reactive and GOFAIR planning concurrently. Rather than mounting a second computer onboard the robot, the GOFAIR planner is implemented on a SUN workstation in the lab. This introduces a communication gap between the reactive and GOFAIR planners. This is currently overcome using a floppy disk. The GOFAIR planner writes the set of subgoals to a 3.5" diskette which is then transferred by hand to the reactive control system onboard the computer. Thus it is not currently possible to have the GOFAIR planner monitor the performance of

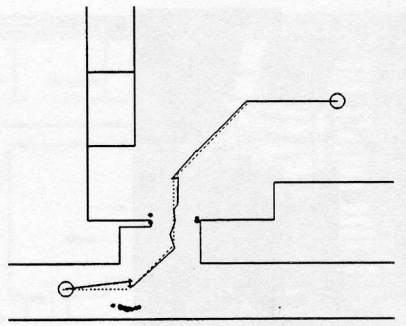

Figure 8: Section of map showing planned and actual paths of robot. The robot starts in the upper right hand corner and moves to the lower left hand corner of the display.

the reactive control system.

Figure 8 shows the GOFAIR planned path and reactive path executed by the robot as it moves out of the robotics lab and into the hallway through the doorway. The planned path (shown as straight lines) is augmented by the reactive planner. The robot proceeds more or less uneventfully from the upper-right corner until it approaches the doorway (the door itself is not shown and constitutes another problem altogether - see the section on future work below). At this point, the *avoid* behaviour is triggered (the small circles denote sonar hits which triggered the behaviour) by the edges of the doorway. These cause small deviations in the robot's path, but the robot detects the doorway on its peripheral sonar sensors only and correspondingly turns away only slightly. Also of note in the diagram are the occasional radical changes of direction near turning points in the planned path. The low-level architecture does not consider the path as a whole, and hence only knows that it must within a specific distance from the desired goal. Thus, if it is outside of that distance (or if it overshoots the goal without realizing it), it still attempts to turn towards and reach that goal with as much precision as it is instructed to use.

When the robot moves outside the door into the hall it encounters an unmodelled obstacle (shown as small circles below the robots path) and executed an *escape* behavior as it detected that it was moving directly towards an obstacle. (This obstacle was actually part of a railing in the hallway.)

4 Discussion and Future Work

This work has demonstrated that it is possible to augment the classical GOFAIR path planners with a reactive control structure to obtain a path planner that is useful for real life mobile robotic tasks. The GOFAIR planner provides a well understood mechanism for planning global paths from one place to another within a mapped environment. The reactive control system modifies the path to take unplanned, unexpected or unmapped obstacles into account. Of course, there is much work still to do on both the GOFAIR and the reactive planners. In terms of the reactive phase, one area of improvement might come about in the subsumption architecture itself. More sophisticated controlling behaviours may be implemented to better handle obstacles. The trade-off here would be to increase the complexity and running-time of the low-level driver in the hope that the enhanced reactive system would choose better paths when avoiding dynamic objects. Increasing the complexity of the subsumption architecture would increase the necessary computing power required to drive the robot, or will result in an increase in the latency time involved in the reactive system

One element that is lacking in the GOFAIR planner is that there is no model of the elevation of floors in the environment. In the map displayed in Figure 2 there is an unlabelled ramp, which can cause serious problems when navigating the robot. In Figure 2 the ramp lies on the only route that can be taken from the robot's home to other portions of the building. An improvement would be to assign a cost to this area, yet allow travel across it if other routes fail. In other words, the path-planner should not treat this route equally, but try to avoid it.

In modelling construction of the world, the question of *doors* is left open. Currently, doors are not modelled in the GOFAIR planner nor in the reactive architecture. In the GOFAIR planner, they are ignored altogether, and they are left to the reactive architecture which will try to avoid them if they are closed (they are simply obstacles) and pass through them if they are open. Both systems could be improved in that the GOFAIR planner could consider doors not as a obstacles, but it could model them and pass this information to the reactive system. The reactive architecture could benefit from this information through the use of a *door-transit* behaviour. In the current implementation in which doors are modelled as always being open, then if the robot encounters a closed door along its desired

path, it will end up blocked as it continually tries to find a path through the door. If the door is opened, then the robot may be able to pass through (unless the door opens towards it, in which case an unsuspecting human might open the door directly into the robot and possibly cause serious damage). If the robot modelled and was able to recognize doors in its static world, then if it detected that the door was closed then it would be able to wait a safe distance away until the door was opened or plan a different path to the goal which does not use this door. Not only would this be more safe, it also cuts down on the power that the robot would normally consume vainly trying to find an opening.

References

[1] R. Brooks. A robust layered control system for a mobile robot. *IEEE Trans. on Robotics and Auto.*, 2(1):14–23, 1986.

[2] C. Fennema, A. Hanson, E. Riseman, J. R. Beveridge, and R. Kumar. Model-directed mobile robot navigation. *IEEE Trans. on Sys. Man and Cyber.*, 20(6):1352–1369, 1990.

[3] Y. K. Hwang and H. Ahuja. A potential field approach to path planning. *IEEE Trans. on Robotics and Auto.*, 8(1):23–32, 1992.

[4] P. Jasiobedzki, M. Jenkin, E. Milios, B. Down, and J. Tsotsos. Laser eye - a new 3d sensor for active vision. In *SPIE Vol. 2059, Sensor Fusion VI*, pages 316–321, Boston, 1993.

[5] M. Jenkin, E. Milios, P. Jasiobedzki, N. Bains, and K. Tran. Global navigation for ARK. In *IEEE/RSJ Int. Conf. on Robotics and Intel. Sys. (IROS)*, pages 2165–2171, Yokohama, Japan, 1993.

[6] J. Jones and A. Flynn. *Mobile robots: inspiration to implementation.* A. K. Peters, Wellesley, MA, 1993.

[7] B. Kuipers and T. Levitt. Navigation and mapping in large-scale space. *AI Magazine*, pages 25–43, 1988.

[8] J. J. Leonard and H. F. Durrant-Whyte. *Directed sonar sensing for mobile robot navigation.* Kluwer Academic Publishers, 1992.

[9] I. Masaki, editor. *Vision-based vehicle guidance.* Springer-Verlag, New York, 1992.

[10] S. B. Nickerson, D. Lond, M. Jenkin, E. Milios, B. Down, P. Jasiobedzki, A. Jepson, D. Terzopoulos, J. Tsotsos, D. Wilkes, N. Bains, and K. Tran. ARK: Autonomous navigation of a mobile robot in a known environment. In *IAS-3*, pages 288–296, 1993.

[11] N. J. Nilsson. *Principles of Artificial Intelligence.* Tioga Publishing Co., Palo Alto, CA, 1980.

[12] A. Zelinsky. A mobile robot exploration algorithm. *IEEE PAMI*, 8(6):707–717, 1992.

Acknowledgments

Funding for this work was provided, in part, by the ARK (Autonomous Robot for a Known environment) Project, which received its funding from PRECARN Associates Inc., the Department of Industry, Science, and Technology Canada, the National Research Council of Canada, Technology Ontario, Ontario Hydro, and Atomic Energy of Canada Limited. The authors gratefully acknowledge the financial support of NSERC Canada.