RobotAssist - a Platform for Human Robot Interaction Research

Nathan Kirchner[1], Alen Alempijevic[1],
Sonja Caraian[1], Robert Fitch[2], Daniel Hordern[1], Gibson Hu[1], Gavin Paul[1]
David Richards[1], Surya P. N. Singh[2] and Stephen Webb[1]
ARC Centre of Excellence for Autonomous Systems
[1]Mechatronics and Intelligent Systems Group
University of Technology, Sydney, NSW Australia
{n.kirchner, a.alempijevic, s.caraian, d.hordern, g.hu, g.paul, d.richards, s.webb}@cas.edu.au
[2]Australian Centre for Field Robotics (ACFR)
University of Sydney, NSW Australia
{rfitch, spns}@acfr.usyd.edu.au

Abstract

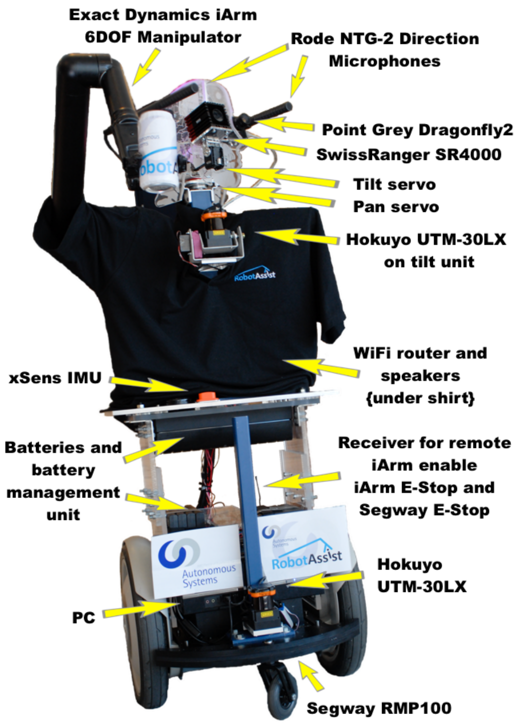
This paper presents RobotAssist, a robotic platform designed for use in human robot interaction research and for entry into Robocup@Home competition. The core autonomy of the system is implemented as a component based software framework that allows for integration of operating system independent components, is designed to be expandable and integrates several layers of reasoning. The approaches taken to develop the core capabilities of the platform are described, namely: path planning in a social context, Simultaneous Localisation and Mapping (SLAM), human cue sensing and perception, manipulatable object detection and manipulation.

1 Introduction

The RobotAssist team consists of researchers from the Australian Research Council Centre of Excellence for Autonomous Systems (CAS) which is a collaboration between the University of New South Wales, University of Sydney, and University of Technology, Sydney. The centre has the overall goal to understand and develop the fundamental robotics science enabling pervasive and ubiquitous application of autonomous systems in broad areas of society. CAS researchers have participated in the RoboCup Rescue and RoboCup Four-Legged and Standard Platform leagues since 1999. The RoboCup@Home league was formed with the aim of motivating and supporting the development of technologies that will enable service and assistive robots to become ubiquitous in society. The RoboCup@Home league (`http://www.ai.rug.nl/robocupathome/`, currently the largest international annual competition for autonomous service robots, uses a realistic non-standardised home environment setting as the arena in which to evaluate robots' capabilities and performance against a set of benchmark tests. The underlying domains tested include: Human Robot Interaction (HRI) and cooperation, navigation and mapping in dynamic environments, computer vision and object recognition under natural lighting conditions, object manipulation, adaptive behaviours, behaviour integration and system integration.

Domestic assistive robots are intended to work alongside humans as aids. One of the research challenges is how to combine autonomous operation with advanced human robot interaction [Plger *et al.*, 2008] [Stuckler and Behnke, 2009] [Kawamura and Iskarous, 1994] [Kawamura *et al.*, 1996]. In 2010 CAS developed a platform for HRI research (known as the RobotAssist Platform) and a series of algorithms for human-robot interaction. These were showcased during that year's RoboCup@Home league. At its core, such a robot needs a sufficient understanding of the world in which it exists in order to facilitate deliberate and successful robot-world interactions. The RobotAssist Platform builds on the navigation, object recognition and planning capabilities we developed as part of our previous efforts in rescue and standard platform leagues [Sheh *et al.*, 2009] and on the outcomes of our recent research on the modes of interaction and communication between humans and robots, identifying moving and stationary humans, methods to improve the perception of human behaviour, participating safely in environments that are shared with humans and morphologies that involve close physical cooperation between humans and robots, such as shared handling and manipulation of objects.

This paper presents a component based framework that allows for integration of operating system independent components, which is designed to be expandable and integrates several layers of reasoning, and the approaches taken to develop the core capabilities of the platform. The breakdown of this paper is as follows: Section 2 and 3 details the system hardware and software infrastructure respectively. Section 4 describes our approach to path planning in a social context while our sys-



Exact Dynamics iArm 6DOF Manipulator
Rode NTG-2 Direction Microphones
Point Grey Dragonfly2
SwissRanger SR4000
Tilt servo
Pan servo
Hokuyo UTM-30LX on tilt unit
WiFi router and speakers {under shirt}
xSens IMU
Batteries and battery management unit
Receiver for remote iArm enable
iArm E-Stop and Segway E-Stop
Hokuyo UTM-30LX
PC
Segway RMP100

Figure 1: The RobotAssist Platform

tem's Simultaneous Localisation and Mapping (SLAM) component is discussed in Section 5. Section 6 details our approach to human cue sensing and perception. Following this, Section 7 focuses on our work in manipulatable object detection and manipulation. Conclusions and future work are discussed in Section 8.

2 Hardware

The RobotAssist platform, depicted in Fig. 1, is built from readily available hardware and open source software for ease of replication by other robotics groups. The design choices during the development of the physical platform where guided by the over-arching capabilities that were deemed essential. Specifically, the platform hardware was selected so that robot would be capable of: mobility in typical office/home environments (corridors, doorways, flat surfaces, soft inclines, etc.), manipulation of typical objects (cups, bottles/cans, etc.) and providing rich multi-modal sensing. A small sensor footprint was essential, for weight and space constraints. The following sub-sections detail the hardware devices that constitute the RobotAssist platform and selection rationale for each.

2.1 Actuation

In order the physically operate in and/or interact with the world the platform needs actuation capabilities. In particular, the capabilities of mobility, manipulation and active sensing. For mobility we selected the well developed and mature Segway RMP100 (`http://rmp.segway.com/rmp-100/`)for its: footprint, differential steering (enabling rotation on the spot), payload capabilities, stability (with optional casters fitted), drive speed, interface (CAN) and its API.

For manipulation we again opted to use a well developed and mature device, an Exact Dynamics iArm (`http://www.exactdynamics.nl/site/?page=iarm`). The iArm was selected for its: of freedom (6DOF), kinematic structure (anthropomorphic), its payload-compliance ratio (the arm is specifically designed to be used in human machine interaction), interface (CAN) and its API.

Finally, the chest sensor mount of the RobotAssist platform is articulated (1DOF - tilt) by a single Dynamixel RX-28 servo and the head (2DOF - pan+tilt) by two Dynamixel RX-28 servos. The intention is to allow the sensor package to be moved quasi-independently from the platform in order to facilitate active sensing. The Dynamixels were selected for the torque rating and interface (RS485).

2.2 Sensors

As is evident in Fig. 1, the RobotAssist platform is instrumented with a number of sensors: two Laser Rangefinders Hokuyo UTM-30LX, Point Grey Dragonfly2 Camera, xSens MTi Inertial Measurement Units (IMU), SwissRanger SR4000 3D ToF Depth Camera, two Rode NTG-2 Directional Microphones and a Plantronics CS60 USB wireless headset GN9330.

The Hokuyo UTM-30LX is used as the primary mapping sensor and was selected primarily for its range (30m) which is necessary for the targeted environments. In addition the small footprint (approximately 100mm x 100mm) and scan rate (40Hz) enable us to use the UTM-30LX in two manners: 1) tilted down at 45° in order to detected table tops during mapping (table legs are often thin and reflective) and, 2) as a sweeping laser (tilting in discrete increments and taking a scan at each) in order to produce a dense 3D point cloud for use in scene interpretation.

The Point Grey Dragonfly 2 Camera is used primarily for face and object recognition tasks (using Haar, SURF, SIFT, etc.). This camera was selected for its sufficient resolution (greater than 640 by 480 @15Hz) and footprint (approximately 30mm x 80mm).

The xSens MTi IMU is used to determine changes in yaw (odometery errors during yaw are typically significant with the Segway) during locomotion, required by both the mapping and localisation components. The xSens MTi IMU was selected for the footprint (approximately 30mm x 60mm) and proven performance on a number of previous platforms [Sheh *et al.*, 2009].

The SwissRange SR4000 produces high frequency (5Hz) 3D point cloud scene representations as well as an intensity image. The SR4000 was selected due to its: scan rate (5Hz), field of view (176 x 144 pixels) and range (approximately 6m).

Finally, the Rode NTG-2 Directional Microphones and a Plantronics CS60 USB wireless headset GN9330, both used to acquire audio streams from the world, were both selected for their limited and well defined sensitivity regions. There is a considerable amount of ambient noise in real-world environments, thus a directed sensing approach to capturing audio is desirable.

3 Software Infrastructure

Future intelligent systems will need to be goal directed and adaptive, able to program themselves automatically by sensing and acting, and be able to accumulate knowledge over their lifetime. Thus, our research goal is to develop general purpose intelligent systems that can learn and be taught to perform many different tasks autonomously by interacting with their environment.

Drawing from our previous experience in large scale systems design [Upcroft *et al.*, 2007], our software is built using a Component-Based Software Engineering (CBSE) paradigm. This approach offers modularity, software reuse, and flexibility in deployment applied to a robotic application. CBSE means that algorithms for perception, navigation and mapping are available as a set of components. Components run asynchronously and exchange information through communication. We use ZeroCs Ice middleware (`http://www.zeroc.com`) extensively in our system for component interface definition, inter-component communication, component deployment, location, activation services, etc. The multi OS and multi programming language support of Ice was crucial for the development of the robotic system, enabling rapid prototyping in MATLAB environments.

We use Orca (`http://orca-robotics.sourceforge.net/`), an open source project that customises Ice to robotic applications and provides an online repository of reusable components. Thus the total number of components that comprise our current system is less than twenty.

In addition we utilise a Black Board system architecture (Mica) for storing any asynchronous information conveyed through human cues [Kadous and Sammut, 2004]. Specifically, a Mob (the primary data structure used in Mica) is constructed describing the cue. The Mob is then sent over TCP/IP to the Mica server, where it is broadcasted to all subscribed components. This architecture allows the system to have some form of continual awareness.

We use four layers of abstraction to reduce the complexity of the problem; a Command layer and three layers of reasoning: Interaction, Planning and Actuation as per Fig. 2.

3.1 Command Layer

The Command Layer interacts with the user via a number of input sensors and processed sensory outputs. This layer implements software modules that are able to recognise when a command has been issued to the robot, distinguishes between speakers by comparing aspects of their voices, and estimate the location of each speaker relative to the robot. Sound signals are processed to reduce the levels of extraneous background noise and isolate the spoken commands of each speaker. In addition to microphones a 3D camera (SwissRanger) is used to acquire information on the immediate environment. Information from the SwissRanger is used to sense when a person is oriented towards the robot and estimate the three dimensional position, direction and movement of a speaker's hands. The sensor information is processed to determine the presence or absence of compound movements, such as gestures indicating that the robot is to follow the speaker, or whether a person is waving to try to issue a command to the robot. Using the auditory input together with user gestures and body language, extracted likelihood estimates are generated for each command within the command list and passed on to the interactive layer.

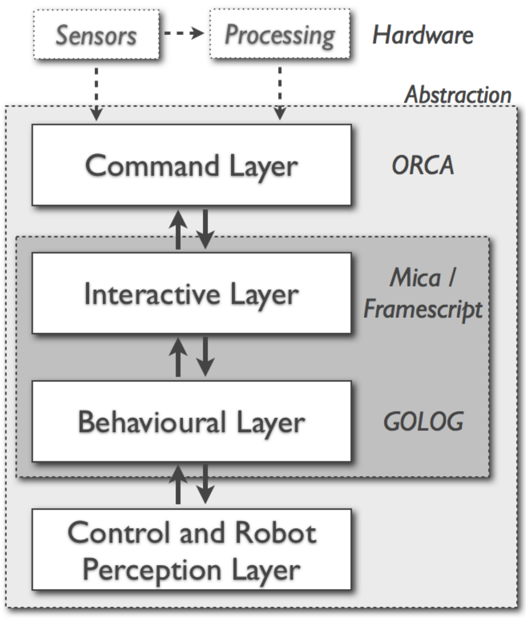


Figure 2: Layers of Reasoning

3.2 Interactive Layer

The Interactive layer uses the contextual information about the world state as well as the internal state of the system when interpreting user commands. This layer involves interpreting multiple, and perhaps conflicting, command inputs in context (robot location, human location, object location, etc.) and decision making (What is being requested? How should it be completed? Is it feasible?). The outputs of the Command Layer are parsed and grouped to estimate their meaning though Mica/Framescript [Kadous and Sammut, 2004]. This layer determines that a person is attempting to issue a command by evaluating the estimates from a predefined list of possible commands encoded in Framescript. A request to confirm the chosen command is performed before the component writes data to the Mica Blackboard as a Task to be acted on by the Behavioural layer. The use of MICA/Framescript provides for a rich conversational interaction between human and robot.

3.3 Behavioural Layer

High level robotic languages overcome numerous deficiencies of traditional programming languages in the context of complex robotics problems [Wobcke *et al.*, 1998]. These provide a layer of abstraction that allows for a variety of programming styles from deliberative constructs that resort to AI planning in order to achieve user goals, through to scripted behaviours when time critical tasks need to be completed. The Behavioural layer employs a high level programming language based on a variant of GOLOG on STRIPS (Stanford Research Institute Problem Solver) [Fikes and Nilsson, 1971]. This component allows for scripted interactions where the required course of action is clear a priori and for deliberative planning where the course of action needs to be considered and elaborated upon or when feedback from the environment is particularly noisy. In this way we satisfy goals using actions that are available by decomposing the task into a series of atomic primitive actions. These series of actions are then supplied to the Control and Perception Layer for execution.

3.4 Control and Robot Perception Layer

The robot needs some understanding of the world in which it is to interact; this layer focuses on deriving such an understanding as well as enacting the supplied actions. Specifically, this component covers building an understanding of the world; (a) environment representation geometry and structure and schematic labelling (kitchen, lounge, etc.), (b) self localisation of the robot in the world (robot base pose, and the arm configuration), (c) location of any humans in the world (pose, heading), (d) objects in the environment (location, size, weight and best grasping position). This layer also addresses actuation; path planning, platform drive control, manipulation methodologies and algorithms required to enact a command once it is understood.

4 Path Planning in a Social Context

Our vision for the path planning component of the system is to produce motions that integrate naturally into a human domestic environment. A simple way to operationalize this concept is to attempt to mimic human locomotion trajectories. Recent studies in analysing human locomotion suggest that to move in a human-like manner, a robot should follow paths with minimal change in curvature so as to minimise jerk [Arechavaleta *et al.*, 2008; Todorov and Jordan, 1998]. Therefore, although the platform has the capability to execute turns with zero linear velocity, we wish to avoid this behaviour. We model the robot as a non-holonomic, car-like system and use a planner that preferentially seeks to minimise change in curvature. In this section, we summarise the algorithm used for planning and discuss its implementation, including a simple error recovery strategy.

4.1 Motion Planning

The algorithm used for motion planning is the well-known Latombe Grid-Search algorithm [Barraquand and Latombe, 1993; Latombe, 1991; C. *et al.*, 2005]. Its name unfortunately seems to imply a discrete search space, but in fact the algorithm does search and represent points in configuration space in continuous coordinates. For convenience, we summarise Latombe's algorithm here.

The robot is modelled as a disc, and controlled by linear and angular velocity commands. Obstacles in the world are represented as a 2D discrete traversability map. The planner accepts a start position and goal position in global coordinates and returns a trajectory represented as a sequence of velocity commands. Each velocity command has an associated time interval. The returned trajectory can thus be executed directly.

The planner builds a *search tree* in configuration space. Its basic operation is to expand a search node by applying a given velocity command for a fixed time interval. The resulting configuration is computed via forward integration. A fixed set of commands is available for expansion. The planner operates by choosing a node from the tree, expanding it, and adding child nodes to the tree. The leaves of the tree (nodes that have not yet been expanded) are stored in a priority queue. After expansion, a node is removed from the queue. If a node chosen for expansion is too 'close' to another node in the tree, it is pruned. If the chosen node is close to the goal, the algorithm terminates and returns the root-to-leaf path. Proximity is determined using an occupancy grid data structure that discretises the configuration space ($\Re^2 \times S^1$) into cells of uniform size. A node is considered *close* to another node if it falls within the same grid cell. Note that continuous coordinates are maintained; the discretisation is used only for efficiently computing proximity. Therefore, the resolution of the grid determines the density of the search tree.

The planner is complete with respect to the resolution of its proximity grid and the time interval of the path set [Barraquand and Latombe, 1993]. In other words, the planner is guaranteed to find a path as long as these parameters are chosen sufficiently small. Because this proof is not constructive, however, we do not have a method for determining parameter values analytically. We hand-tuned them empirically and found a reasonable grid resolution of $0.2m \times 0.2m \times \frac{\pi}{8}rad$ and path set time interval of 2 seconds. Our path set has angular velocities chosen from $\{-\frac{\pi}{4}, -\frac{\pi}{16}, -\frac{\pi}{32}, 0, \frac{\pi}{32}, \frac{\pi}{16}, \frac{\pi}{4}\}$ and linear velocities from $\{0.4m/s, 0.2m/s\}$. Our priority queue used a cost function that combines minimum distance to goal with minimum change in curvature.

Implementing this algorithm requires handling two special cases. The planner moves the robot near the goal, but an additional method is required to achieve a desired distance tolerance. We implemented a PD controller for this 'end-game' case. Secondly, we allow turning-in-place only at the start position, when the robot is already stationary. To implement this, we add zero-linear-velocity elements to the path set when expanding the root node. The path is executed in open-loop fashion with fixed frequency replanning. Because the robot is moving quite slowly (under 0.35 m/s) we re-plan every 2 seconds (0.5 Hz).

4.2 Error Recovery

Our system is subject to uncertainty in control and sensing. The result of sensing uncertainty is the presence of errors in the traversability map. The result of control uncertainty is that the robot's actual trajectory differs from its predicted trajectory. A source of this control uncertainty is variation in mass distribution due to changes in manipulator position or payload. Both of these factors can lead to the robot moving closer to obstacles than desired. The system must recover from both expected and unexpected situations where it finds itself too close to an obstacle.

The error detection and recovery problem is not the focus of our current research, but to ensure a level of safety and liveness we chose to implement a basic strategy involving two components inspired by the state of the art in autonomous systems [Urmson *et al.*, 2008]. Our goal was to avoid catastrophic failure where the platform halts and does not attempt to recover. The first component is a 'panic' system that continuously checks raw range data. If an obstacle is too close, the system reacts and stops the platform. The second component attempts error recovery and rotates the robot in place until a new plan is found by the planner. If the planner continues to fail even with no obstacles immediately in front of the robot, error recovery continues by commanding a small linear velocity. This random motion in free space continues until a valid plan is available.

5 SLAM

Simultaneous localisation and mapping (SLAM) [?] is the problem where a mobile robot needs to build a map of its environments and simultaneously use the map to locate itself. For RobotAssist, although it is operating in a roughly known environment, some detailed position of the obstacles (such as furniture) may not be available. In the @home competition, each team is given limited time to build the map either manually or autonomously. We choose to build the map fully autonomously using the laser sensors. One horizontal 2D laser mounted at the bottom of the robot and one tilted 2D laser looking at around 45° down/forward.

5.1 Pose-only SLAM from 2D laser

The 2D laser scan based map is built using only horizontal laser scans. Here we adopt the popular pose-only SLAM strategy where SLAM is divided up into two components, front end and back end [?]. In front end SLAM, the goal is to find relationships between robot poses. In our model a robot pose corresponds to the time a laser scan is taken. The relationship is calculated using Iterative Closest Point (ICP) [?]. The algorithm is performed on both consecutive scans as well as poses where loop closure occurs. Since scanning can be done at 40Hz, we need to become selective in choosing our scans. By checking the odometery given by the wheel encoders, we can set a threshold on how far the robot has moved or rotated to determine when a robot pose should be added to the map.

The most challenging part of this 2D SLAM is the reliable detection of loop closure. One major problem is that the two laser scans coming from different angles cannot be matched easily even if they are scanning the same environment. After extensive analysis and testing, we decided to use a simple strategy to obtain loop closure information accurately and reliably. After finding a point when potential loop closure could occur we deliberately ask the robot to perform a 360 degree turn on the spot, this will make sure that at least a couple of scans are generated from the similar position and will guarantee that ICP will be able to get the correct relative poses. This method can be considered an active SLAM implementation.

During back end SLAM for robot pose optimisation we use a non-linear least squares approach [?]. Since

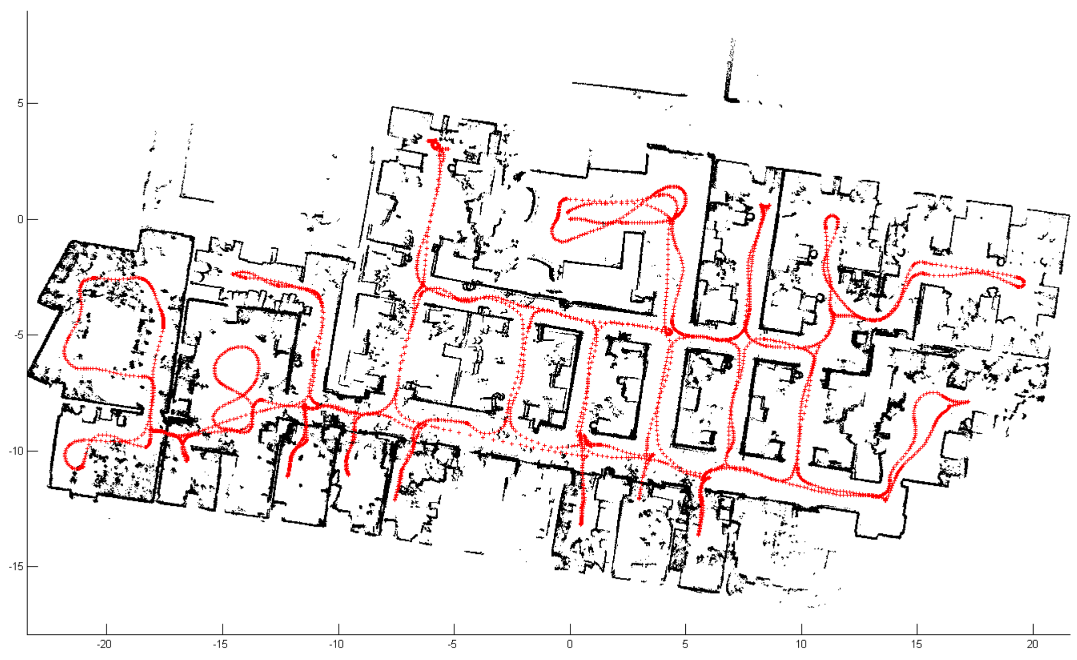


Figure 3: Pose-Only SLAM Map of CAS

the Jacobians involved are all sparse matrices, the optimisation problem can be transformed into the solving of sparse linear equations, which can be performed efficiently using sparse linear algebra. Here the C++ suite sparse library is used. By optimising all our positions and correlating the laser scans associated with these positions we can generate accurate occupancy grid maps.

The 2D SLAM algorithms has been tested extensively, Fig. 3 is a map generated in this way for the CAS-UTS area.

5.2 Obstacle detection using the top laser

Although the 2D horizontal laser can be used to generate a good quality map, some obstacles such as tables may not be detected (the horizontal laser can only detect the legs of the table). Thus we use another laser looking down/forward to help build a active traversable map. The scans from this laser are translated onto a 2D plane and correlated with an updated set of robot poses. By fusing the the obstacles map with the SLAM map, a traversable map is obtained Fig. 4.

6 Human Cues

Furthermore, we have developed algorithms for determining if the head/body position is oriented towards the robot. Intensity images from the SwissRanger are used for detecting the presence of a face while the range image is used to determine the head pose, shown in Fig. 5. Using this data with an anthropomorphic model we are able to robustly detect the observed humans hands and, through a priori gesture meaning knowledge, determine the humans intention and/or decipher gestures, shown in Fig. 6. Once the sensed data is interpreted and perceived as human cue, the information is made available to the system.

7 Manipulation

The capabilities of assistive robots, such as the RobotAssist platform, can be greatly extended by incorporating an anthropomorphic manipulator. A manipulator is a robotic actuator which can be controlled to perform tasks that include the grasping and manipulation of objects/tools, and the safe transfer of objects between

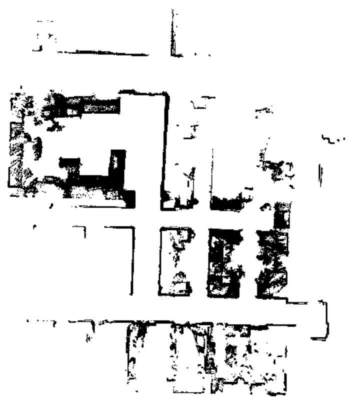


Figure 4: Fused Traversable Map

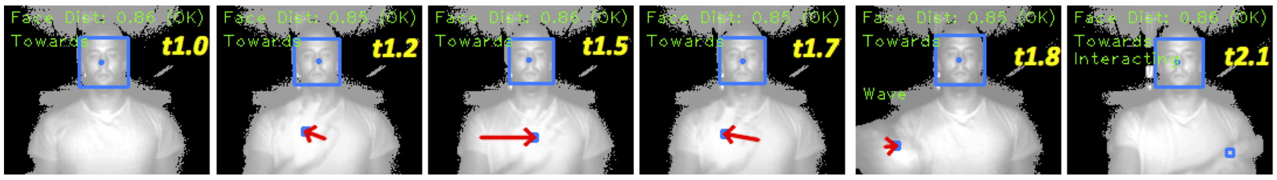


Figure 6: Behavioural sequence where the human instantiates an interaction by waving while looking at the robot

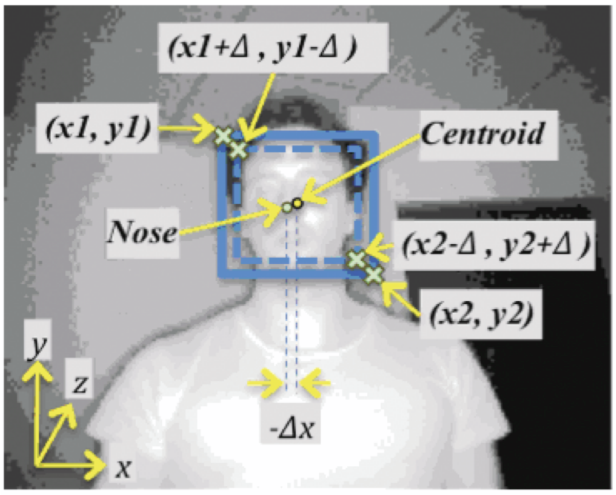


Figure 5: Detecting the face and head pose

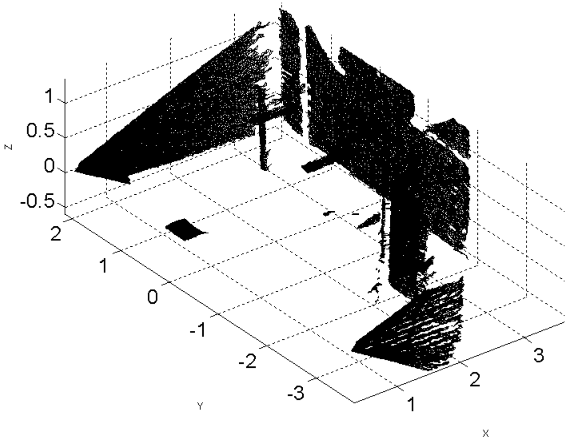


Figure 7: Sweeping the laser range scanner through 30° and the point cloud map (over 100k 3D points) that is generated.

the robot and a human. The manipulator used on the RobotAssist platform is the Exact Dynamics iArm6DOF manipulator.

Manipulation tasks are challenging since they need to be performed in close proximity with humans, through complex, natural environments, which may be initially unknown. Therefore, a mobile manipulator system must have the following four integrated capabilities: sensing and mapping of the surrounding environment which may contain obstacles; detection of graspable objects; pose selection to determine the joint configuration of the manipulator so that the end-effector of the manipulator (i.e. the hand) can grasp the detected object; and collision-free motion planning through the map to a selected pose such that the manipulator can grasp the detected object.

7.1 Environment and Obstacle Detection

There are many techniques to generate a map of the surface geometry for an environment. In the case of the RobotAssist platform, two sensor-based methods have been used. By slowly sweeping a 2D Hokuyo laser range scanner, with a field-of-view of 270°, through a desired angle of rotation it is possible to generate a high resolution point cloud map. An example map generated by sweeping the laser ranger scanner is shown in Fig. 7. The other alternative is to use the SwissRanger which has a faster scan rate (10Hz) but has a narrower field-of-view. In order to fuse the data from multiple scans into a surface, an adaptive “distance field” map representation (volumetric technique) is used. Data fusion is based on the technique proposed by [Curless and Levoy, 1996] and improved by [Webb, 2008] for a real-time and online implementation which is able to handle thin plates and sharp features. The implementation includes a spatial index over multiple signed distance fields implemented as an octree of small 3D grids. This provides a sparse representation to minimise memory usage and an index for efficient updates. The output of the fusion process is a mesh map where vertices (a point cloud) can be rapidly queried for manipulator planning and collision avoidance.

7.2 Object Detection

By using a combination of the SwissRanger and the camera, which are both mounted on the head of RobotAssist, it is possible to search for graspable objects in the environment. The grasping primitives are used to describe the size, position and orientation of a graspable object, such that the end-effector of the manipulator can be orientated correctly, and to ensure that the object can be successful grasped. The first step of the object detection process is the object-background segmentation. To achieve this, feature primitives are extracted from a camera and geometric discriminators derived from 3D point data generated by a SwissRanger depth camera. The feature primitives are extracted from a modified SURF implementation. Following this, orthogonal-axis planner histograms and geometric discriminators enable target objects to be robustly detected in, and segmented from, cluttered environments. Fig. 8 shows RobotAssist performing a search for a graspable object. Once identified and isolated from the environment, the 3D point data corresponding to the object is utilised to determine the grasping primitives of the object.

7.3 Pose Selection

To successfully grasp an object, as shown in Fig. 9, a manipulator pose that corresponds to an adequate grasp primitive must be determined. An optimising pose selection method has been found to be the most effective way of combining the grasping task objectives, the collision avoidance checks, and the self-collision checks. The developed approach uses an adaptation of the Pose Selection using Levenberg-Marquardt (PSuLM) algorithm [Webb, 2008] [Paul *et al.*, 2009]. This generates a near optimal solution by repeatedly applying the algorithm to select a configuration that minimises a cost function. Task constraint cost terms are devised to ensure that the orientation and position of the end-effector will enable the object to be grasped. The cost function design strategy can scale and normalise multiple task constraint cost terms, a self-collision and joint-limit proximity cost term, and an obstacle proximity cost term to create a composite function suitable for use in non-linear optimisation. PSuLM provides a path in configuration space so the inverse kinematics problem does not need to be solved.

7.4 Motion Planning

Finally, in order to move from the current manipulator pose to the desired pose, motion planning for the manipulator is required. Since an assistive mobile manipulator platform is expected to be working in close proximity to humans, safe motions are vital. In this implementation, the environment is assumed to be static in the instant between when a map is generated and when a motion is planned then executed. A safe motion requires the manipulator only passes through known free space. That is, for any motion: the manipulator must never be in collision with the known obstacles in the map, it must not enter space which is unknown, and it must not self collide with any part of the RobotAssist platform. A modified bi-directional Rapidly-exploring Random Tree (RRT) is applied to this part of the planning. The RRT modification involves projecting the randomly sampled configurations onto a constraint satisfying sub-space. The Levenberg-Marquardt algorithm is once again utilised to minimise a dynamically generated cost function. The manipulator will thus be safely manoeuvred to the pose where the object in the natural environment can be grasped.

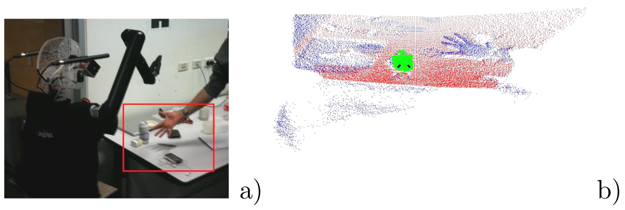


Figure 8: a) RobotAssist platform searching for graspable objects. b) A single SwissRanger scan with the graspable object (i.e. a can) detected and highlighted.

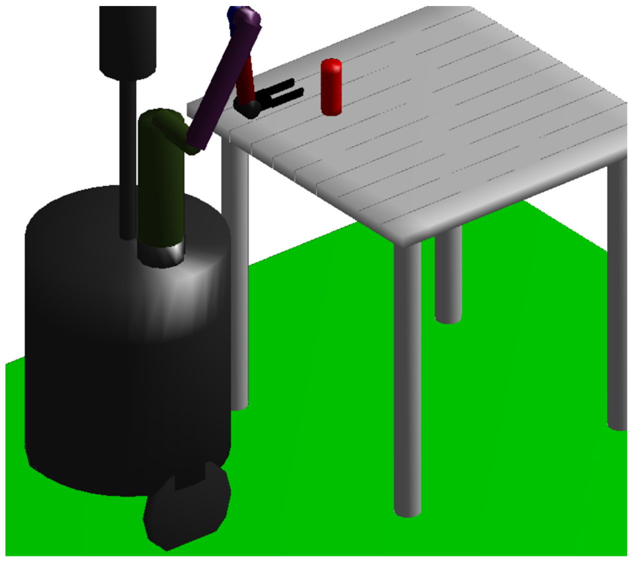


Figure 9: The model of the manipulator that is used for pose selection, and the model of the graspable object which has been detected.

8 Field Testing

As previously mentioned, the RobotAssist platform was entered in the RoboCup@Home league of the 2010 RoboCup in Singapore. As this was our inaugural entry into the @Home league, the practical realities of the task requirements of the competitions were not fully known or appreciated a priori. In such, this experience proved to be an excellent testing ground for the RobotAssist platform: the Hardware, Software Infrastructure and Core Capabilities - each of which will now be discussed in turn.

8.1 Hardware

In general the selected hardware preformed as expected, observations of note were made during the competition. For instance, even though we were able to successfully autonomously navigate through the typical home environment presented in the competition, there were significant challenges to mobility (narrow passages, doorways, clutter, etc.) that resulted in slow traversal (the platform expended a significant amount of time adjusting the RMP100's bearing). It was observed that platforms with fewer holonomic constraints were able to transverse the arena in less time as those platforms did not expend comparatively large amounts of time adjusting their bearing.

Another interesting observation of note was that most teams elected to use low DOF (2-4 DOF) manipulators and utilise the 3DOFs of their platform base to facilitate manipulation. In practice however, it seemed that those teams spent a considerable amount of time adjusting the base pose and sensing, and then re-adjusting and re-sensing, prior to manipulation. Where as our platform, with the the redundancy that the 6DOF arm introduces, was able to manipulate objects with considerably higher success and in considerably less time.

The 1-DOF articulated chest sensor mount proved to be of little value as a sweeping laser (the time taken was too costly) but when angled down was of considerable use finding furniture (tables, chairs and the like) during mapping. The head articulation also proved valuable for base-independent active sensing. As previously mentioned, re-orientating the RMP-100 incurred a significant time cost where as searching utilising the DOFs of the head was comparatively cheap.

8.2 Software Infrastructure

The usefulness of our component based software infrastructure was evident during the preparation leading up to each of the competition tasks. For example, our component based infrastructure allowed us to continue fundamentally re-arranging the operation (the high level planning) of the robot, based on observations/performance, up until moments before our trials began. Furthermore, the concurrent-access for multi-developers that our infrastructure provides proved to be highly beneficial as it allowed multiple team members to concurrently conduct on-line development and testing of different system components/capabilities on the single shared platform.

8.3 Core Capabilities

The following observations of note were made of the core capabilities during the competition. In general the approach to path planning was successful, however, it was evident that the completion time of navigation is a significant contributor to the overall success of robots in the competition. SLAM needs to be robust to frequent occurrences of dynamic objects such as people, moving furniture and other robots. Any algorithms designed to be human 'aware' must be able to discriminate between the target human and the audience or capable of robustly tracking the target human. Finally, manipulation that incorporates the platform's base DOFs as non-redundant DOFs suffer in environments with limited access.

9 Conclusions

This paper presented a component based framework. The framework: enables integration of operating system independent components, is designed to be expandable and integrates several layers of reasoning. Further, this paper outlined the approaches taken to develop the core capabilities of the platform, namely: path planning in a social context, Simultaneous Localisation and Mapping (SLAM), human cue sensing and perception and manipulatable object detection and manipulation.

As demonstrated through our experiences during field testing (detailed in the previous section), our platform, with its component based system infrastructure, proved to be flexible, effective and supportive of multi-developer concurrent-access.

Future work will focus on developing a wider set of core capabilities and adapting the system architecture to more effective facilitate run-time component debugging and monitoring.

Acknowledgments

This work is supported by the Australian Research Council (ARC) Centre of Excellence programme, funded by the ARC and the New South Wales (NSW) State Government, and by RobotAssist and the University of Technology, Sydney.

References

[Arechavaleta *et al.*, 2008] G. Arechavaleta, J.-P. Laumond, H. Hicheur, and A. Berthoz. An Optimality Principle Governing Human Walking. *IEEE Trans. on Robotics*, 24(1):5 –14, feb. 2008.

[Barraquand and Latombe, 1993] J. Barraquand and J. C. Latombe. Nonholonomic multibody mobile robots: Controllability and motion planning in the presence of obstacles. *Algorithmica*, 10:121–155, 1993.

[C. *et al.*, 2005] Howie C., K. M. Lynch, S. Hutchinson, G. A. Kantor, Wolfram B., L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations.* MIT Press, Cambridge, MA, June 2005.

[Curless and Levoy, 1996] B. Curless and M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In *Computer graphics proceedings, annual conference series*, volume 2006, pages 303–312, New Orleans, 1996. Association for Computing Machinery SIGGRAPH.

[Fikes and Nilsson, 1971] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3-4):189–208, 1971.

[Kadous and Sammut, 2004] M. W. Kadous and C. Sammut. Mica: Pervasive middleware for learning, sharing and talking. In *Proc. IEEE PerCOM Workshops*, pages 176–180, 2004.

[Kawamura and Iskarous, 1994] K. Kawamura and M. Iskarous. Trends in service robots for the disabled and the elderly. volume 3, pages 1647 –1654 vol.3, sep. 1994.

[Kawamura *et al.*, 1996] K. Kawamura, R.T. Pack, M. Bishay, and M. Iskarous. Design philosophy for service robots. *Robotics and Autonomous Systems*, 18(1-2):109 – 116, 1996.

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

[Paul *et al.*, 2009] G. Paul, N. Kirchner, D. K. Liu, and G. Dissanayake. An Effective Exploration Approach to Simultaneous Mapping and Surface Material-type Identification of Complex 3D Environments. *Journal of Field Robotics, Special Issue on Three-Dimensional Mapping*, 26(11-12 SI):915–933, 2009.

[Plger *et al.*, 2008] P.-G. Plger, K. Pervlz, C. Mies, P. Eyerich, M. Brenner, and B. Nebel. The DESIRE service robotics initiative. 2008.

[Sheh *et al.*, 2009] R. K. Sheh, A. Milstein, M. J. McGill, R. Salleh, B. Hengst, and C. Sammut. Semi-Autonomous Robots for RoboCup Rescue. In *Proc. ACRA-09*, pages 1–10, 2009.

[Stuckler and Behnke, 2009] J. Stuckler and S. Behnke. Integrating indoor mobility, object manipulation, and intuitive interaction for domestic service tasks. pages 506 –513, dec. 2009.

[Todorov and Jordan, 1998] E. Todorov and M. I. Jordan. Smoothness Maximization Along a Predefined Path Accurately Predicts the Speed Profiles of Complex Arm Movements. *J Neurophysiol*, 80(2):696–714, 1998.

[Upcroft *et al.*, 2007] B. Upcroft, A. Makarenko, M. Moser, A. Alempijevic, A. Donikian, W. Uther, and R. Fitch. Empirical Evaluation of an Autonomous Vehicle in an Urban Environment. *JACIC*, 4(12):1086–1107, 2007.

[Urmson *et al.*, 2008] C. Urmson, J. Anhalt, D. Bagnell, C. R. Baker, R. Bittner, M. N. Clark, J. M. Dolan, D. Duggins, T. Galatali, C. Geyer, M. Gittleman, S. Harbaugh, M. Hebert, T. M. Howard, S. Kolski, A. Kelly, M. Likhachev, M. McNaughton, N. Miller, K. Peterson, B. Pilnick, R. Rajkumar, P. E. Rybski, B. Salesky, Y. Seo, S. Singh, J. Snider, A. Stentz, W. Whittaker, Z. Wolkowicki, J. Ziglar, H. Bae, T. Brown, D. Demitrish, B. Litkouhi, J. Nickolaou, V. Sadekar, W. Zhang, J. Struble, M. Taylor, M. Darms, and D. Ferguson. Autonomous Driving in Urban Environments: Boss and the Urban Challenge. *J. Field Robotics*, 25(8):425–466, 2008.

[Webb, 2008] S. S. Webb. *Belief Driven Autonomous Manipulator Pose Selection for Less Controlled Environments*. PhD thesis, University of New South Wales Australia, 2008.

[Wobcke *et al.*, 1998] W. Wobcke, M. Pagnucco, and C.i Zhang. Agents and Multi-Agent Systems Formalisms, Methodologies, and Applications, Based on the AI'97 Workshops on Commonsense Reasoning, Intelligent Agents, and Distributed Artificial Intelligence. 1441, 1998.