ISR: An Intelligent Service Robot

M. Andersson, A. Orebäck, M. Lindström, and H. I. Christensen

Centre for Autonomous Systems,
Royal Institute of Technology
S-100 44 Stockholm, Sweden
WWW home page: `http://www.cas.kth.se`

Abstract. A major challenge in mobile robotics is integration of methods into operational autonomous systems. Construction of such systems requires use of methods from perception, control engineering, software engineering, mathematical modelling, and artificial intelligence. In this paper it is described how such a variety of methods have been integrated to provide an autonomous service robot system that can carry out fetch-and-carry type missions. The system integrates sensory information from sonars, laser scanner, and computer vision to allow navigation and human-computer interaction through the use of a hybrid deliberative architecture. The paper presents the underlying objectives, the underlying architecture, and the needed behaviours. Throughout the paper, examples from real-world evaluation of the system are presented.

1 Introduction

Service robotics is an area of research that is rapidly expanding. We strongly believe that we will have small robots roaming around in our houses in the near future. An excellent example of such a device is the autonomous Electrolux TriLobote vacuum-cleaner, that was revealed to the public during spring 1998. The application potential of robots is enormous, ranging from boring tasks like vacuuming, to advanced household tasks such as cleaning up after a dinner party. Recent progress, particularly in sensor-based intelligent robotics, has paved the way for such domestic robots. It is however characteristic that relatively few robots are in daily use anywhere, and very few mobile robot systems are being mass produced. Examples of mass produced mobile systems include the HelpMate Robotics platform for delivery of food and x-ray plates at hospitals, and the RoboKent floor sweeper produced by the Kent Corporation. Both have only been produced in relatively small series (in the order of hundreds).

The primary obstacles to the deployment of robots in domestic and commercial settings, are flexibility and robustness. The robots must be flexible so that they are relatively easy to deploy under different conditions, and so that they can be used by non-experts. This requires a rich set of control functions, combined with an intuitive user interface and automatic task acquisition functions (like automatic learning). Robustness, in terms of sensory perception, is needed to allow for operation 365 days a year under different environmental conditions.

Typically, research robots have been demonstrated in a single environment under ideal working conditions, like an in-door environment with artificial light (and no windows). Such systems provide a proof of concept, but there is a long way from such systems to commercially viable systems. Some might claim that the gap is purely a development process with no or very little research involved. We claim that there is a need for fundamentally new methods to empower deployment of robots in everyday settings, and there is thus a need for fundamentally new research to enable use in the above mentioned market segments.

To pursue research on robust, flexible, and easy-to-use robot systems for everyday environments, an intelligent service robot project has been initiated at the Centre for Autonomous Systems at KTH. The long-term goal of the project is deployment of an intelligent robotic assistant in a regular home. The system must be able to perform fetch-and-carry operations for the human operator. To accommodate such tasks, it must be able to understand commands from a non-expert in robotics. This requires an intelligent dialogue with the user. Having received an instruction, the robot must be able to plan a sequence of actions and subsequently execute these actions to carry out the task. In a realistic scenario, the robot will encounter unexpected events such as closed doors and obstacles. To be perceived as a useful appliance, the robot must cope with such ambiguities in an 'intelligent' manner. To perform fetch and carry missions, which include opening of doors, picking up and delivery of objects, etc, the robot must be equipped with actuators that allow for manipulation. A basic functionality for such a robotic system is the ability to perform robust navigation in a realistic in-door environment. The first phase of the *Intelligent Service Robot* project has thus been devoted to the development of a flexible and scalable navigation system, that can carry out simple tasks like delivery of mail in an office environment, that includes rooms similar to a regular living room.

In this paper we describe the results of this initial phase of the project. The robot is equipped with a speech and gesture interface for human computer interaction. For navigation, a combination of ultra-sonic ranging and laser-based ranging is used. The different navigational functions are implemented as a set of behaviours that provide direct coupling between sensory input and actuator control. The output from different behaviours are integrated using simple superposition. To control the execution of a mission, a state manager, and a planner are used. For the integration of the overall system, a hybrid deliberative architecture has been developed, which allows for easy integration of the different system components.

Initially, related research on service robotics and navigation in an in-door environment is reviewed in Section 2. We then outline the overall architecture of the system in Section 4. Each of the components in the system are described in the following sections to give an impression of the complexity and diversity of the overall system. A number of issues related to the implementation of the system are reviewed in Section 8. The system has been used in a large number of experiments in laboratory and living room settings. Some of the results from these experiments are provided in Section 9. Finally a summary and issues of future research are provided in Section 10.

2 Related work

The area of service robotics has recently received significant attention. The area is, however, closely related to general in-door navigation, where one of the first efforts was the Stanford cart by Moravec [1, 2]. The area gained significantly in popularity with the change from sense-plan-act type systems to reactive behaviour-based systems, as motivated by Brooks subsumption [3] and Arkin's AuRA system [4].

In more recent time, some of the most dominating efforts have been those of Simmons in terms of his Xavier system [5], Reactive Action Packages by Firby [6, 7], the RHINO system from Bonn [8], the ROMAN system from Munich [9], and the Minerva [10] and Sage [11] systems from CMU.

The Xavier system was specifically designed for autonomous navigation in an in-door setting. The system has been designed to carry out point-to-point missions, navigating between known places in an in-door environment. It is well-known that sensory perception is non-robust, and a principal issue has thus been to provide the needed functionality to cope with such uncertainty. Xavier is built around a hybrid deliberative architecture. A topological map of the environment is used for initial planning of missions, while obstacle handling is based on potential field methods. The system includes Partially Observable Markov Processes (POMPS) for managing uncertainty. The system has been reported to have a success rate of more than 90% for missions, that together cumulates to a total travel length of several kilometres.

Based on the successful use of Markov models in Xavier, the team at University of Bonn has built a museum tour guiding system [8]. The system has a probabilistic map of the environment that allows for automatic localisation and error recovery in structured environments. The system is also based on a hybrid deliberative architecture that allows for automatic handling of obstacles, like humans, in its immediate proximity. The RHINO system has later been updated and recoded for the Minerva system [10] that gave tours at the American Museum of Natural History in New York. Overall, the probabilistic framework has turned out to be very successful. All of the above systems rely on ultra-sonic sensing as the primary modality for localisation and navigation.

The RAP (Reactive Action Packages) framework developed by Firby [6, 7] has been used in a number of different robot systems. The RAP system does situation specific action selection. Based on sensory interpretation of the context, an index is generated that enables action selection. The method relies heavily on explicit internal models of the environment, but in such situations it is a very powerful and intuitive framework for robot control. The methodology is accompanied by a framework for programming of reactive systems. The system has been used extensively at University of Chicago for robots like the Animate Agent [12]. This system includes colour based vision for obstacle detection, gesture interpretation and object recognition. Robust performance is reported for a laboratory setting without windows.

In Munich the ROMAN (RObot MANipulator) system has been developed over the last ten years [9]. This is a system that performs fetch and carry missions in a laboratory setting. The system includes a manipulator for interaction with the environment. The system uses a laser scanner for localisation, and sonar for detection of obstacles. The system relies on vision for recognition of objects (and obstacles), servoing on natural landmarks like doors, and manipulation of objects. The system is based on a sense-plan-act type of architecture. Planning is a key component to facilitate interaction with the environment for tasks like opening doors or picking up objects in a drawer. For interaction with objects, the system relies heavily on a priori specified geometric models. The system has been demonstrated in a laboratory setting, where is it able to navigate and perform error recovery in the presence of dynamic obstacles like humans. Error handling is here performed using a plan library that explicitly encodes all of the situations to be handled.

The Sage system developed by Nourbakhsh et al. [11] at CMU is derived from ideas originating in the Dervish system [13]. The original Dervish system uses the layout of an office building to detect symmetries, that allow relative localisation in combination with odometry. The system is controlled explicitly by a planner, based on local map information. In this sense, the system is highly deliberative and only obstacle avoidance, using potential fields, is performed reactively. The system uses map abstraction, rather than behavioural abstraction, for control, which is a departure from the present trend towards behavioural systems. The Sage system has replaced the planner by pure sequential/imperative programming of missions in combination with artificial landmarks, for navigation in the Dinosaur Hall at the Carnegie Museum of Natural History. A unique characteristic of this system is that it is fully autonomous, in the sense that it does automatic docking for recharging, and it can thus be used for extended periods of time without human intervention.

All of the above systems illustrate how robotics gradually has reached a stage where in-door navigation in natural environments is within reach. Many of the systems rely on prior defined maps, but when such maps are available, the systems can perform autonomous navigation. A few of them also include manipulation of objects. The trend has been towards hybrid deliberative architectures, where low level behaviours are used for control, while the actual invocation and configuration is carried out by a supervisor, that in turn receives missions specifications from a planner. This is also the approach that has been adopted in the system described in this paper.

3 Goals

The intelligent service robot demonstrator is intended to show that it is possible to build a useful robot for a house or office environment. The system therefore has to meet realistic demands concerning task complexity, human-robot interaction and the capabilities of acting in a realistic environment.

The specification of the operating environment for the robot has been performed in a very pragmatic way. By looking at homes and our own laboratory environment (mainly offices), specifications of room sizes, floor types, lighting conditions, etc have been made. To be able to test the robot in a realistic environment, our main laboratory has been turned into a living room, furnished with sofas, tables, book shelves, etc., see Figure 1.

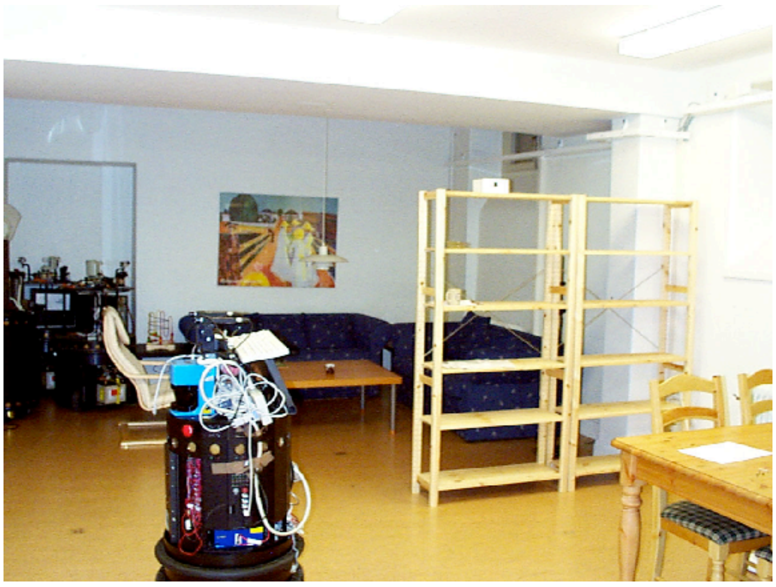

Fig. 1. The living room. Our main lab has been turned into a living room in which there are sofas, tables, book-shelves, and so on, to make it possible to test the robot in a realistic setting.

We believe that in a typical home or office, changing the environment for the sake of the robot will either be too costly or undesirable for other reasons. Therefore, our intentions are for the robot to work in an unmodified environment without artificial landmarks. The goal is to have the robot navigate safely, combining data from several sensors, and using a minimum of a priori information about its environment.

To be useful, the robot must also be able to manipulate different objects. The work is focused on recognition of everyday objects like cups, books, soda cans, etc., but it is equally important to be able to open doors, drawers etc. as this will be necessary in a home or an office. The challenge is to be able to do this in an unstructured and partly changing environment where objects positions are unknown or only partially known. Manipulation work is not described in this paper as it has only recently been started.

An important skill of home or office robots is the ability to communicate with its operator or coworker. Therefore a study of the human-robot interface is performed within the project. The interface includes speech and gesture recognition. Both of these communication modalities are natural for humans and therefore easy to learn for a non-specialist. A more detailed description is given in Section 5.2.

There are a variety of possible tasks for a domestic or office robot. The tasks we are currently considering are mainly fetch-and-carry tasks like

- Go to the refrigerator and bring back the milk.
- Deliver mail to a person.
- Setting and cleaning a table.
- Riding an elevator.

4 System Architecture

Most software architectures for mobile robots are layered. By layers, we mean distinct parts that have different levels of authority. The distinction is never totally clear in a modern robot system; a resulting motion of a robot is normally the consequence of a complex command sequence. It is seldom a direct path from the top (highest authority) layers to the motors in the lowest layer. However, a system description can be easier to explain if we divide it into layers.

We have chosen to divide our explanation of the system into three layers; the *deliberate layer*, the *task execution layer*, and the *reactive layer*. The deliberate layer makes deliberation decisions according to the state of the system. These can be derived from robot objectives or orders from a human user. The task execution layer makes sure that the plans of the deliberate layer are carried out. The last layer consists of sensor, behaviour, and actuator modules that can be configured and connected/fused together in a flexible network. The task execution layer will decide on the configuration of the network, that will solve the task at hand. This describes a hybrid architecture design of the selection paradigm. In other words, planning is viewed as configuration, according to the definition by Agre and Chapman [14]. An overview of the layers in the architecture can be seen in Figure 2, where each part is described in more detail below.

4.1 Deliberate Layer

The deliberate layer consists of a *planner* and a *human robot interface* (HRI). The HRI interprets human commands and intentions using speech, gestures, and keyboard as input. The commands are relayed to the planner that makes a plan for their execution.

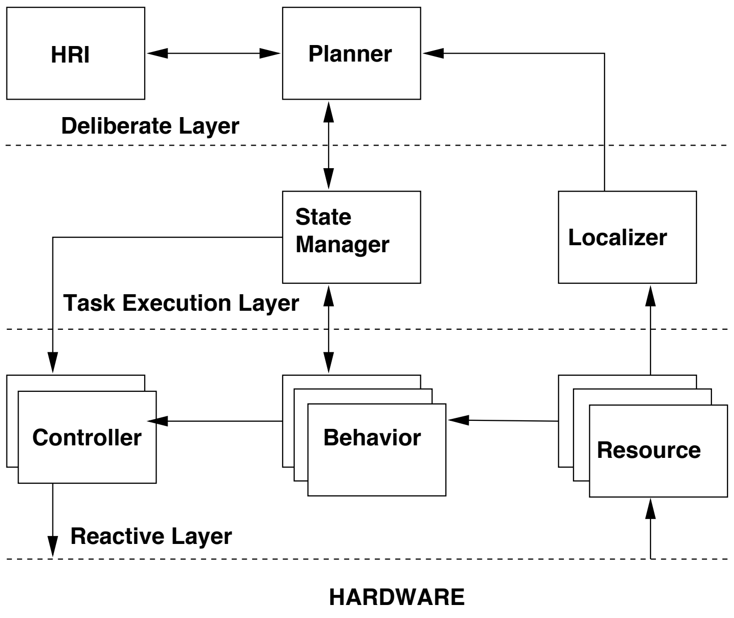

Fig. 2. Layers of the system architecture

4.2 Task Execution Layer

The task execution layer consists of a *state manager* and a *localiser*. The state manager configures the modules in the reactive layer to solve the task. The configuration is at the moment done using a lookup table for construction of the appropriate network. The localiser keeps track of the current position and provides metric and topological map data. These data are also used by the planner for path planning.

4.3 Reactive Layer

The reactive layer is closest to the hardware. It consists of a large set of modules that can be configured into a network connecting the sensors and actuators of the robot. The network defines tight sensorimotor connections, which results in fast reflexive behaviours without any involvement of deliberation.

The modules of the reactive layer are of three types: *resources*, *behaviours*, and *controllers*. The resources read and preprocess sensory data to extract essential information, which is forwarded to the behaviours. The behaviours act as mappings between sensing and actuation. They present propositions for the control of the actuators, depending on the data from one or more resources. Finally, the controllers, the modules closest to the actuators, fuse the propositions of the behaviours to a unified control of one or more actuators.

5 Deliberation

5.1 Planning

The planner belongs to the deliberate layer of the robot system architecture. It receives orders from the human-robot interface or from keyboard input. The planner will, depending on the current state of the robot, accomplish the given orders. During execution, the planner will give information about the state of the robot to the user. This consists of current action, success or failure to execute it, the position, and the immediate route.

The planner has access to a topological map (provided by a map server) with information about important nodes in the environment and metric distances between nodes. The topological map is augmented by an estimate of the travelling time between nodes. These time intervals are not necessary proportional to the actual metric distance, since the area in between the nodes can be cluttered with obstacles that the robot have to spend time to swirl. When a destination is given to the planner it will plan the route that has the shortest time estimate to completion. The plan is created using Dijkstra's algorithm for directed graph search. The route will normally contain a number of intermediate nodes. The robot traverses the nodes in the given order, while avoiding obstacles. The estimate in travelling time is updated when the path between two nodes has been traversed. Initial estimates of the travelling time are presently set to a time proportional to the metric distance.

5.2 Human-Robot Communication

A service robot working in a house or office environment will need a user-friendly, safe, simple-to-use and simple-to-learn human-robot interface (HRI). The most important reason is that the robot is going to work together with non-experts as opposed to most industrial robots of today. Possible ways of communicating with the robot include keyboards, touch-screens, joy-sticks, voice, and gesture commands. The preferred modes depend upon a combination of environment, user skill, task, and cost. In a noisy environment keyboard input is probably preferred over voice input, while in a situation where the operator needs to use his/her hands for other tasks, voice input is a better choice. However, for the environments and applications discussed within the intelligent service robot project, we are focusing our research efforts on a user interface combining both speech and gesture recognition. These modes of communication are natural for humans and complement each other well.

An overview of the current system is shown in Figure 3. The input is provided through a camera and a wireless microphone. The camera is connected to the gesture recognition module, which will search the images for gestures. The base unit of the microphone is connected to the speech recognition program, that will analyse and convert the spoken words into text. Speech and gestures are then combined into fully working commands in the command constructor. If a user says "Go there", a pointing gesture can be used to recognise in which direction to go and the result of this is, e.g., the command “Go left”, which is then forwarded to the planner. The robot is also equipped with a speech synthesiser, which today is primarily used for acknowledgement by repeating the input command. In the future, the speech capabilities will be used in a more dialogue based communication with the end-user, similar to [15].

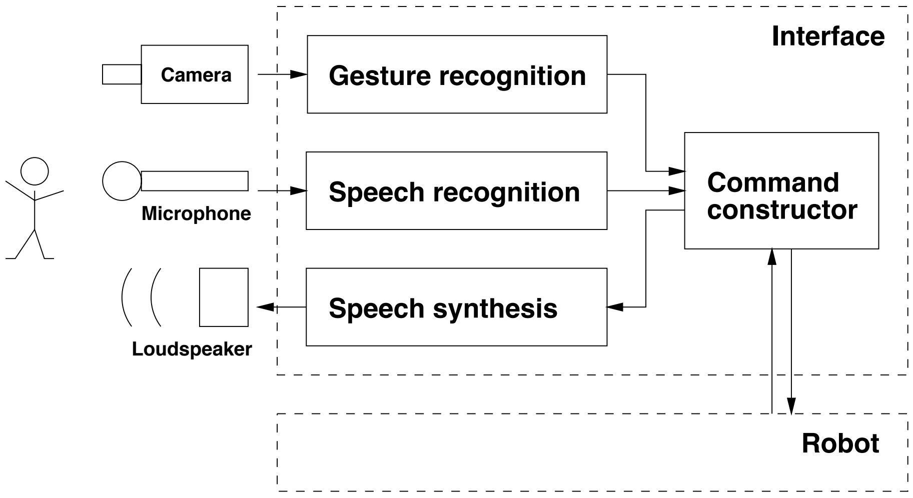

Fig. 3. Overview of the human-robot interface.

The language interpretor of the robot is built around a finite state automata framework. The valid voice commands are defined by the grammar listed below. Capital words are the actual spoken words (terminals). Lower case words are groups of possible words or sequences (non-terminals). Words in curly brackets are optional. All voice commands given in a session are described by a `voice` sequence.

```
voice:
      ATTENTION {command-sequence} IDLE {voice}
command-sequence:
      ROBOT command {command-sequence}
command:
      STOP
      EXPLORE
      HOW ARE YOU DOING TODAY
      FIND THE BOX
      FOLLOW ME
      DELIVER MAIL IN {THE} absolute
      LOCATE IN {THE} absolute
      GO relative
      GO TO {THE} absolute
relative:
      BACKWARD
      FORWARD
      LEFT
      RIGHT
      THAT WAY
absolute:
      OLLES ROOM
      DINNER TABLE
      LAB
      LIVING ROOM
      MAILBOX
      SCIENTIFIC AMERICAN
```

The current gesture recognition module is able to find pointing gestures, that allows it to identify if a person is pointing left, right, or not at all. The gesture is recognised through identification of the relation between the head and the hands. Skin colour segmentation is used for identification and tracking of the head and the hands. Recognition of skin colour is not stable enough, there are many objects that have similar colours. Therefore other cues need to be integrated into the system and recently *depth from stereo*, has been incorporated in order to increase performance. We have so far not used the gesture interpretation in real tasks, but a probable scenario could be the following. A person is standing in front of the camera, telling the robot to *Pick up that*, and pointing at an object in the scene. The robot then identifies the spoken command, finds out that it also needs gesture input and uses the image information to identify which object the operator is pointing at. It is important that the gesture and speech interpretation is done in parallel as the operator needs to be pointing and talking concurrently.

6 Mission Execution

The Mission Execution layer consists of two parts, a *state manager* and a *localiser*. These two components are described in below.

6.1 The State Manager

The state manager is a finite state automaton that controls the reactive layer. At startup, the state manager reads a file that associates states and behaviours. This file can be reread during execution, to allow for online reconfiguration. Behaviours that are not connected to a controller, are controlled directly by the state manager.

A new task is initiated by a message from the planner to the state manager. This message contains a *state* and *data*. The state constitutes a certain set of behaviours that should be run in ensemble. An example of a state is *GoPoint* which translates into the behaviours *Gopoint* and *Obstacle Avoidance*. The data is usually a goal point, but for some states, such as *Explore*, the data field is irrelevant. Each behaviour is associated with a controller. Upon a change of state, the state manager informs the controllers which behaviours it should integrate. A task-list may also be supplied by the user through a file when the system is debugged without a planner.

The state manager awaits messages from the behaviours that tell whether the task was successfully carried out or if it failed. In the case of success, the planner is informed and a new command may be issued. Otherwise the state manager goes into a Stop-state, which is also sent to the controllers. A backup-state may defined for each state. If a failure occurs, this backup-state is initiated if it exists, otherwise the planner is notified. At any time a new command from the planner may be received and execution is then pre-emptied or modified. Especially the command to stop is given high priority. The state manager will also forward any goal point updates originating from the localiser.

6.2 Localisation

One of the most important capabilities of a robot is to know where it is. The knowledge of the current location can be expressed in many different ways. Here, we will say that the robot is localised if it knows where in a room it is (metric information). We have used two different approaches to localisation, one using sonar sensors and one using a laser range finder.

In the sonar approach the robot initially has a topological map of the environment. To be able to get metric information it uses its sonar sensors. The basic steps in the localisation process are the following:

– Let the robot explore a room to collect sonar data. The robot can either automatically perform the exploration or an operator can use the joystick to guide the robot. The data is used to automatically create a sonar landmark map. During this process it is also possible for an operator to assign names to different places in the room (goal points).
– When a room is revisited, the robot collects new sonar data, again by exploring the room, and these are matched to landmarks in the original map. From this mapping, the position and the orientation of the robot is estimated.
– Following the previous step, the robot iteratively updates its current position estimate.

The process is described below, for more details see [16].

Sonar data using triangulation Our Nomad 200 robot, used for most of our experiments, has a ring of 16 ultrasonic sensors. Sonars are known to provide noisy data. The sound sent out by the sonar is spread in a cone emerging from the source (actually the process is even more complicated). The emitted sound is then reflected and the time difference between emission and reception can be used to calculate the distance to the object that reflected the sound. Due to the spreading of the emitted sound, the reflecting point(s) can be anywhere on a surface. To remedy the problem, a triangulation technique has been developed, which fuses sonar readings from multiple positions. The basic idea is that if two sonar readings come from the same object, the position of that object must be in the intersection of the two cones. By accumulating such intersection points, it is possible to use simple voting to identify the most stable landmarks.

Map acquisition The robot is started at an arbitrary position in a room. This position will become its reference point in the room. It then moves around, collecting sonar data using the triangulation scheme outlined above. Some of the collected sonar data will be stable over an extended time interval. By selecting the most stable data combined with readings from the odometry, it is possible to create a landmark map of the room. The landmarks are selected automatically by the robot itself. However, when studying the data a posteriori, it is evident that corners on bookshelves, tables, chairs and door posts are likely to become landmarks. At present, the 20 most stable landmarks are stored for each room. During this process, it is also possible for an operator to name different positions, called goal points, in the room. The goal points are symbolic references that can be used by the user when giving commands to the robot.

This map making process is repeated in each room. In larger rooms, for example a long corridor, the limited number of landmarks is inadequate. Such rooms are divided into a number of smaller regions that each have a separate set of landmarks.

Map matching When the robot revisits a room, it will use the previously stored map to localise itself. The topology map, used by the planner, is used to determine switches between rooms. Upon entering a room the robot has the option to carry out a relocalisation process. If the robot encounters an error or looses track of its position it can perform an absolute localisation in a room. The robot carries out exploration in the room and collects a set of at least 10 landmarks. The set of collected landmarks are then matched against the stored map, and the associated rigid transformation (translation and rotation) is estimated based on the best possible match between the two maps.

6.3 Localisation using a laser range finder

The robot also has an on-board laser scanner. The sensor is a SICK PLS-200 scanner, that provides a 180° scan of its environment, represented as 361 measurements in the range 0.7 m – 100 m and with an accuracy of 5 cm. In this particular work, each room is represented as a rectangle (represented by width and length). The laser scan is then matched against the room model and an estimate of the position is feed into a Kalman filter that maintains an estimate of the robot position. Given an initial estimate of the position, the following steps are taken, see also Figure 6:

1. Using odometry, the last position estimate is updated. This is the time update phase for the Kalman filter.
2. The updated position estimate is used in combination with the estimated uncertainty, to generate a set of validation gates that allows filtering of data, to reject outliers, as shown in Figure 7.
3. The filtered data are feed into a range weighted Hough transformation. Using an estimate of the position of walls it is now possible to perform model based segmentation.
4. The estimated Hough lines are used for definition of more accurate validation gates, that allow segmentation of data points into segments.
5. A least square line fitting is performed on the segmented data points.
6. A rigid transformation between fitted line segments and the model is used for updating of the Kalman filter. This is the measurement update step of the Kalman filter.

For a more detailed description of the procedure see [17] and [18].

7 Reactive Control

The reactive control layer consists of resources, behaviours, and controllers.

A resource is a server that distributes data to client behaviours. The data typically originates from a sensor, such as sonars, cameras, etc. A resource can also be a client to another resource, and compute higher level information.

A behaviour is a transformation between perception and action. Each behaviour in the system delivers proposals for the control of actuators. Depending on the function of the behaviour, it can use a number of resources to obtain the necessary information for the decision making. In view of the client/server concept, a behaviour is a server with controllers as clients. It is also a client with respect to resource-servers.

In this system, there are basically two types of behaviours. One type, e.g., the behaviour *GoPoint*, receives a goal-value (typically a goal point) from the state manager and reports back success or failure. The other type requires no goal-value. Consequently, it never reports to the state manager. One example of this type of behaviour is *AvoidObstacle*.

The controllers are responsible for sending direct commands to the robot's actuators. They will fuse the output from a set of behaviours defined by the state manager. This behaviour fusion mechanism has to be specified in the implementation of the controller. It could for example be an arbitration or a weighted vector summation of the proposals from the behaviours. Other schemes use fuzzy logic or voting.

Upon a change of state, the controller receives new directives from the state-manager that specifies which behaviours the controller should listen to.

The set of behaviours implemented in the system is outlined below.

7.1 GoPoint

The *GoPoint* behaviour steers towards a certain (x,y)-position in space. The odometry is used to determine the steering angle needed to reach the position.

7.2 Avoid

The *Avoid* behaviour detects obstacles and provides control to avoid them. The sonars are used for obstacle detection. Avoid uses a protection shield. It will suggest the robot to move away from any obstacle entering this shield. One parameter, the forward protection radius, specifies how close obstacles are allowed to come in front of the robot. A second parameter, the side protection radius, specifies how close obstacles are allowed to come to the side of the robot. The fusion of *GoPoint* and *Avoid* will drive the robot smoothly around small obstacles.

7.3 Explore

The *Explore* behaviour moves the robot away from positions it has visited before. If the robot for instance starts within a room, this behaviour will try to make the robot visit every open space in the room. When the behaviour believes there are no places left to visit, it will clear it's memory and start over again. This behaviour uses the sonar resource. Explore together with Avoid, is used during the localisation phase. When the localiser reports that enough landmarks have been found, the explore behaviour is terminated.

7.4 MailDocking

The *MailDocking* behaviour is used to pick up mail from a mail slot. It assumes that the robot is positioned exactly at the mailbox, where the forklift is used to pick up a mail-tray at a specified height. Currently, pure dead reckoning is used, but will later be done using visual servoing.

7.5 DoorTraverser

The *DoorTraverser* behaviour takes the robot through a doorway. The position of the doorway is fed into the behaviour by the state manager. The behaviour itself is incorporating avoidance facilities that have been trimmed for the purpose of narrow doorway traversal. DoorTraverser can tolerate corrupted doorway position information. In the current state, the behaviour can find the doorway if the doorway position error is less than 1.5 meters. Two versions of DoorTraverser have been implemented. One using sonars and another using laser range data.

7.6 FollowPerson

The *FollowPerson* behaviour will locomote the robot towards a human. At the moment, the human must face the robot while it follows. Monocular vision is normally used to detect and track the skin-colour of the head of the person. The *Avoid* behaviour is normally used simultaneously, so that the robot does not run into the person. The behaviour is described in more detail in [19].

7.7 CornerDocking

The *CornerDocking* behaviour uses the laser range finder to position itself relative to a corner. The method deployed is to extract the two walls defining the corner. Based on an estimate of the corner position from the laser data, a proportional control scheme is used to servo the robot to the desired position. The behaviour requires that the corner is in the "field of view" of the laser scanner when it is initiated.

7.8 Behaviour Fusion

The data received from the behaviours is fused or arbitrated by the controller to give a control signal to the actuators. Upon receiving the command to stop from the state manager, the controller stops the robot, disconnects the behaviours and enters an idle mode.

The controller implemented so far, is a vehicle controller. Future work includes integration of an arm controller, when the system is ported to our Nomad XR4000. The vehicle controller controls the steering and drive motors of the robot. Two formats of input data from the behaviours have been implemented.

In the first method, each behaviour provides a histogram containing 72 cells. The index of each cell represents an absolute direction from the robot centre, thus resulting in an angular resolution of five degrees (this is similar to a vector field histogram). The value of each cell indicates how much the behaviour wants to travel in that direction. The histograms are fused by component-wise summation and smoothed by convolution with a normalised, truncated, and discretised Gaussian function g. The resulting fused histogram is given by

$$f = (\sum h_j) * g,$$

where the discrete convolution with the Gaussian is performed on the circular buffer of cells. The robot will set course in the direction of the maximum value of f.

The second implemented data format is a polar representation (magnitude and direction). The vector outputs from the behaviours are fused by vector summation according to

$$v = \sum v_j,$$

where v_j is the output from behaviour j. The robot will head in the direction of the sum vector v. Both formats also include a speed proposal for the robot.

In order to make the robot drive along a smooth trajectory around larger obstacles, two schemes have been investigated. The first consists of exchanging the GoPoint behaviour with what we call a PathGoPoint behaviour. This uses sonar-data to construct a grid map used to compute a path around obstacles. In the other method, the vehicle controller detects when the sum of a behaviour fusion is close to zero. The controller will then start up another behaviour called Swirl. This behaviour will produce a steering angle, perpendicular to the object, to be added in the fusion process. Details on our strategy for smooth control can be found in [20].

8 Implementation

In this section we discuss the the hardware facilities of the robot, and the actual software implementation structure.

8.1 Hardware

The described system has been implemented on a Nomadic Technologies *Nomad 200* platform. It is a circular robot that is 125 cm high and with a diameter of 53 cm. There is an on-board 133 MHz Pentium computer with 128 MB RAM running Linux OS. The computer can communicate with the the outside world through a radio Ethernet link. Our robot is equipped with a wide range of sensors and actuators. Some of them are displayed in Figure 8.

Sensors The sensors are both of proprioceptive and exteroceptive type. The proprioceptive sensors give the robot it's body awareness, while the exteroceptive sensors sense the world around it. We will describe all exteroceptive sensors in the following paragraphs, see also Figure 8.

Ultrasound Sensors One of the primary sensors on the platform is an omnidirectional ultrasound sonar ring with 16 equiradial Polaroid elements. The sonars can measure distances from 15 cm up to 8 m and are placed approximately 0.8 m above the floor making it possible to detect obstacles at table height even at close range.

Infrared Sensors The infrared sensors are placed in a ring, similar to the sonar ring, but 38 cm above the floor. The measurement range is up to 40 cm. They can be used for last minute detection of obstacles that cannot be detected by the ultrasound sensors, e.g., a low coffee table.

Proximity Laser Scanner On top of the robot body we have mounted a SICK PLS-200 proximity laser scanner. It measures distances in a half-plane, approximately 95 cm above the floor. The sensor provides 361 readings resulting in an angular resolution of 0.5 degrees. The maximum measurement range is 100 m with an accuracy of 5 cm.

Odometry The odometry measures the rotation and translation of the wheels and the turret. The accuracy is high, but due to slippage on the floor etc. the measurements cannot be fully trusted.

Vision system A camera head, consisting of two relatively fixed colour cameras, is mounted on a pan-tilt unit on top of the robot. The cameras are connected to a Fujitsu image processing board.

Actuators The actuators of the robot enables it to locomote and to manipulate. A short description of the actuators are given below.

Wheel Base The wheel base consists of a three wheel synchro-drive, such that they always point in the same direction. The robot has a zero gyro-radius drive which enables it to turn on the spot. The maximum speed of the robot is 61 cm/s and the maximum turning speed is 60 degrees/s.

Turret The turret is mounted on the wheel base with a rotating joint. The joint can be rotated with a maximum turning speed of 90 degrees/s. All the sensors, except odometry, are mounted on the turret.

Lift mechanism A lift mechanism is mounted on the front of the robot. The lift has a gripper that can grip objects that are up to 15 cm wide and weight less than 9 kg. The lift can reach object from the floor and up to a height of 122 cm.

8.2 Software

The objective of the software implementation was to fulfil the following criteria:

– Easy integration of new behaviours
– Easy interfacing of new hardware devices
– Transparent performance on multiple platforms
– Efficient runtime performance
– Simple debugging

The implementation of the software architecture has been performed using an object-oriented framework. We have used C++ to code all parts of the architecture, except for some Nomad supplied code that is written in C. Separate parts of the system, e.g., Resources, Behaviours, Controllers etc., run as separate processes communicating using sockets. This makes it possible to simultaneously run the system on several different platforms. The programming in C++ is done in correspondence with the ANSI standard and the operating system specific parts are wrapped in the software package Adaptive Communication Environment (ACE), to promote portability [21]. The total system is at the moment (December 1998) of approximately 68000 lines of code, including comments.

Process management All the software components described in Section 4 are processes. These are connected to each other in a network using global and local sockets.

The *process manager* keeps track of all processes used in the architecture. It is similar to the function of the ORB in CORBA. Each new process has to inform the process manager of its name, host, and address. It maintains a database for this information as well as information about dependencies between processes.

If a process wants to establish communication with another process, the process manager will provide the necessary address. The process manager will also start processes when they are required.

The process manager has an *executor* daemon on each computer used in the system. On demand from the process manager, an executor will start or kill processes used in the system. The executor will also monitor the processes it has started and report unexpected terminations. The executor can be viewed as a tool for the process manager to reach over all the computers in the system. The executor will always work as a slave for the process manager, except for the instance when the communication to the process manager is broken. In such an event, the executor will kill all the processes it has spawned, and return to an initial state waiting for a process manager to leash it again. Thus, the executor does not have to be restarted.

ACE - Adaptive Communication Environment The adaptive communication environment is an object-oriented toolkit that implements strategic and tactical design patterns, to simplify the development of concurrent, event-driven communication software. It provides a rich category of classes and frameworks that perform communication tasks across a wide range of operating system platforms. It can for example handle event demultiplexing and handler dispatching, interprocess communication, shared memory management, message routing, dynamic configuration of network services, threading, and concurrent control. For more information, see [21].

9 Results

The system presented above has been evaluated in our laboratory, including the living room shown in Figure 1. The evaluation has included experiments on localisation, map building, and fully autonomous mission execution for tasks like mail delivery.

Localisation involving the sonar system is described in detail in [16, 22]. In general, it is possible to localise the system with an accuracy of 5 cm within a room of the laboratory. Using the laser scanner, a continuous updating of ego-position can be achieved with an accuracy of 1-5 cm, depending on the environment. In certain situations it is only possible to detect the wall at the side of the robot, i.e., in a long corridor. During such situations the longitudinal uncertainty grows significantly until a feature like a door becomes visible. The laser results are reported in further detail in [17, 18].

For evaluation of the fully integrated system, a mail delivery scenario has been defined. The robot receives the command "Robot Deliver Mail in *room*" from the speech interface, where *room* is one of several pre-defined rooms. The robot will then plan a path to the mail-slot, drive to the mail slot, pick-up mail and finally drive to the designated room and announce arrival of mail. During this mission it will avoid obstacles and re-localise, if the uncertainty in position grows beyond a certain bound. This task has been carried out more than 100 times. The system has a reliability of about 90%, where the major source of problems is due to loss of localisation, which can be provoked if the robot encounters a large number of obstacles, or drives a long distance without any cues for re-localisation. This might happen in a long corridor with no or few landmarks that allow re-localisation. The laser range based localisation is expected to solve this problem, once it has been fully integrated. Another source of error is the docking to pick-up mail. It is critical that the robot is positioned with an accuracy of a few centimetres, as it otherwise will miss the mail-tray. To achieve this the laser-based corner docking is used. Due to lack of tactile feedback on the fork-lift it is however, possible to miss the tray.

Recently the system was demonstrated at a trade fair, where it navigated for 8 hours, only interrupted by replacement of batteries, every hour. During this period the robot moved about in an area of 20 by 20 meters. The map of the demonstration site was acquired during an explore session and subsequently used for relocalisation, when the errors grow beyond a fixed threshold. The robot performed robustly during the demonstration period.

Overall, it is a robust and versatile system. The robot can with little effort be transferred to a new environment and be taught the layout of the plant using exploration behaviours. Once a topological map and landmarks for the involved rooms have been acquired, it can be sent on missions.

10 Summary

A fully operational robot system has been constructed. The system has been developed using a behaviour-based approach and integrated using a hybrid deliberative architecture. The system includes facilities for spoken and gesture based commanding of the robot. Once instructed to carry out a particular mission, it will generate a plan to complete the mission. The list of tasks to be accomplished is given to a state manager that configures the set of behaviours, and monitors the execution of individual tasks. In the event of errors, a simple recovery strategy is deployed. For localisation, the system utilises sonars and a laser scanner. A map of the environment is automatically constructed using an exploration strategy. The system has been implemented on a Nomad 200 robot from Nomadic Technologies Inc.

The system is capable of performing a variety of tasks like mail delivery and point-to-point based navigation in a natural environment, like a regular living room. Extensive experiments have demonstrated that the system has a very high degree of robustness.

The primary limitations of the system is in terms of robust localisation in the presence of few natural landmarks. In addition, the system has problems when it encounters low obstacles and they frequently will not be detected by the present set of sensors.

The system has been built by a set of 10 graduate students, that each have contributed methods within their particular area of expertise. The basic architecture has been carefully designed to allow for easy (and flexible) integration of new behaviours. Today a "plug-n-play" functionality is available for adding new behaviours, with a minimum need for behaviour designers to know the internal structure of the system.

Presently the system is being ported to a Nomadic Technologies XR4000 robot, that includes an on-board PUMA 560 manipulator. This robot has a richer set of sonars (just above the floor and at 80 cm height), and a better SICK scanner (LMS-200). These facilities will allow for more robust navigation. The new challenge will be integration of manipulation into the system, which introduces a number of interesting research questions in terms of coordination and non-holonomic path planning. The short-term goal is to extend the current scenario to include the ability to open and close doors and to ride in an elevator. The manipulation system also opens up new problems in grasping, using visual servoing, and force-torque sensing.

11 Acknowledgement

The service robot has been constructed by a team consisting of Magnus Andersson, Henrik I Christensen, Magnus Egerstedt, Martin Eriksson, Patric Jensfelt, Danica Kragic, Mattias Lindström, Anders Orebäck, Lars Pettersson, Hedvig Sidenbladh, Dennis Tell, and Olle Wijk. The entire team has been instrumental in this effort. Throughout the effort valuable guidance and advice was received from Ronald C. Arkin, Jan-Olof Eklundh, Bo Wahlberg, and Anders Lindquist.

The Centre for Autonomous Systems is sponsored by the Swedish Foundation for Strategic Research, without this support the effort would never have been possible.

References

1. H. P. Moravec, "Towards automatic visual obstacle avoidance," in *Proceedings of Int. Joint. Conf. on Artificial Intelligence in Cambridge,MA*, p. 584, 1977.
2. H. P. Moravec, "The Stanford cart and the CMU rover," *Proceedings IEEE*, vol. 71, pp. 872 – 884, 1983.
3. R. Brooks, "A hardware retargetable distributed layered architecture for mobile robot control," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 1987.
4. R. C. Arkin, "Integrating behavioral, perceptual, and world knowledge in reactive navigation," in *Robotics and Autonomous Systems, Vol. 6, pp. 105-22*, 1990.
5. R. Simmons, "Structured control for autonomous robots," in *IEEE Transactions on Robotics and Automation*, 1994.

6. J. R. Firby, "Modularity issues in reactive planning," in *Third International Conference on AI Planning Systems*, (Menlo Park, CA), pp. 78–85, AAAI Press, 1996.
7. J. R. Firby, *Adaptive Execution in Complex Dynamic Worlds.* PhD thesis, Computer Science Dept., Yale University, 1989. TR-YALEU/CSD/RR #672.
8. M. Beetz, W. Burgard, A. B. Cremers, and D. Fox, "Active localization for service robot applications," in *Proceedings of the 5th International Symposium on Intelligent Robotic Systems '97*, 1997.
9. U. D. Hanebeck, C. Fischer, and G. Schmidt, "Roman: A mobile robotic assistant for indoor service applications," in *Proceedings of the International Conference on Intelligent Robots An Systems 1997*, 1997.
10. S. Thrun, M. Bennewitz, W. Burgard, A. Cremers, F. Delaert, D. Fox, D. Hahnel, C. Rosenberg, N. Roy, J. Schulte, and D. Schulz, "Minerva: A second generation museum tour-guide robot." CMU Tech Report (url://www.cs.cmu.edu/˜thrun/papers).
11. I. Nourbakhsh, "The failures of a self reliant tour robot with no planner," tech. rep., Carnegie Mellon University, Robotics Institute, url: http://www.cs.cmu.edu/˜illah/SAGE, 1999.
12. J. R. Firby, P. Prokopowicz, and M. J. Swain, *Artificial Intelligence and Mobile Robotics*, ch. The Animate Agent Architecture, pp. 243–275. Menlo Park, CA.: AAAI Press, 1998.
13. I. Nourbakhsh, *Artificial Intelligence and Mobile Robotics*, ch. Dervish: An Office Navigating Robot, pp. 73–90. Menlo Park, CA.: AAAI Press, 1998.
14. P. Agre and D. Chapman, "What are plans for?," *Robotics and Autonomous Systems*, vol. 6, pp. 17–34, 1990.
15. H. Asoh, S. Hayamizu, I. Hara, Y. Motomura, S. Akaho, and T. Matsui, "Sociall embedded leraring of office-conversant robot jijo-2," in *Int. Joint Conf. on Artificial Intell. 1997*, 1997.
16. O. Wijk and H. I. Christensen, "Extraction of natural landmarks and localization using sonars," in *Proceedings of the 6th International Symposium on Intelligent Robotic Systems '98*, 1998.
17. P. Jensfelt and H. I. Christensen, "Laser based position acquisition and tracking in an indoor environment," in *International Symposium on Robotics and Automation - ISRA'98*, (Saltillo, Coahuila, Mexico), December 1998.
18. P. Jensfelt and H. I. Christensen, "Laser based pose tracking," in *International Conference on Robotics and Automation 1999*, (Detroit, MI), May 1999.
19. H. Sidenbladh, D. Kragic, and H. I. Christensen, "A person following behaviour," in *IEEE International Conference on Robotics and Automation 1999*, (Detroit, MI), May 1999. (submitted).
20. M. Egerstedt, X. Hu, and A. Stotsky, "Control of a car-like robot using a dynamic model," in *IEEE International Conference on Robotics and Automation*, May 1998. Accepted for presentation.
21. D. C. Schmidt, "The adaptive communication environment: Object-oriented network programming components for developing client/server applications," in *11th and 12th Sun Users Group Conference*, 1994.
22. O. Wijk, P. Jensfelt, and H. I. Christensen, "Triangulation based fusion of ultrasonic sonar data," in *IEEE Conference on Robotics and Automation*, IEEE, May 1998.

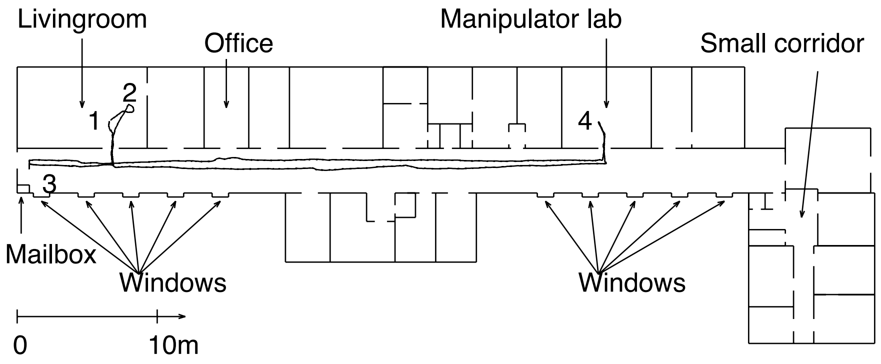

Fig. 4. A layout of the CAS lab floor showing the path of the robot executing some commands. At point 1 the robot is told to localise itself in the living-room. It then wanders around randomly, collecting sonar data and matching these to a previously stored map. At point 2 it knows where it is and stops. It is then told to deliver mail in the manipulator lab, which means that it will go to the mail slots, point 3, pick up the mail, and then go down the hallway to the lab, point 4. Finally it is told to go back to the living room.

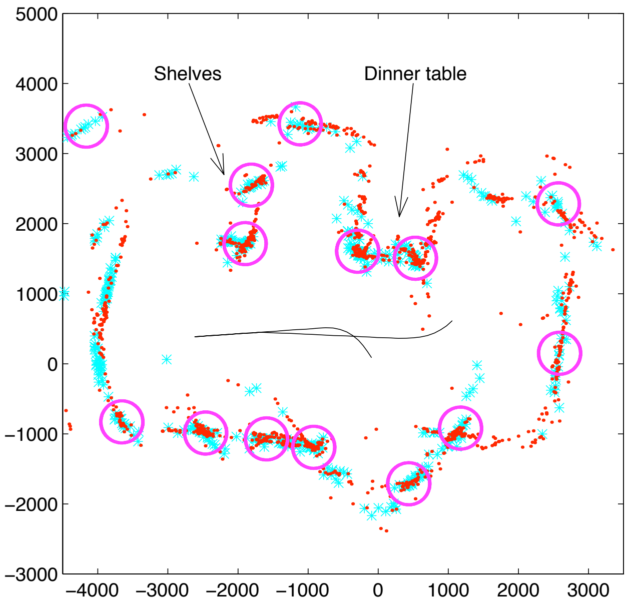

Fig. 5. A sonar map of the living room. The circles indicate where a map landmark has been matched to a point from the current run. In this case 13 landmarks have been matched.

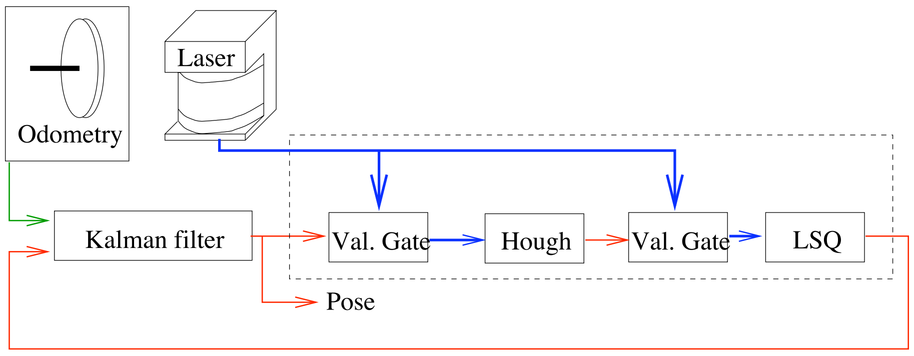

Fig. 6. The laser based localisation method

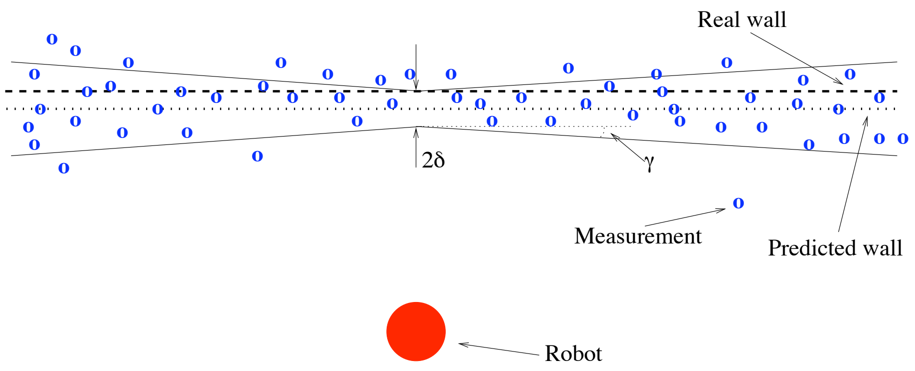

Fig. 7. Selection of reasonable measurements. The concept of a *validation gate* is in this case an area, points outside this area will be rejected.

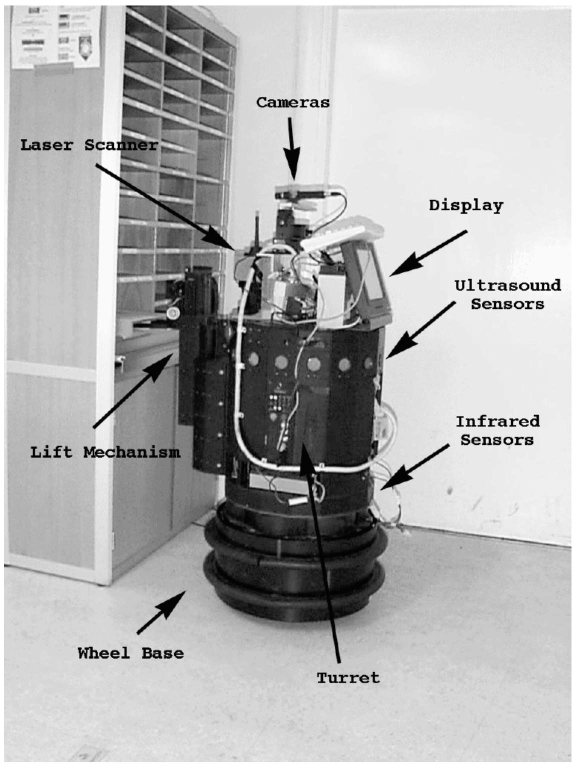

Fig. 8. Hardware Placement.