A Software Control Architecture based on Active Perception for Mobile Robotics[1]

R. Vázquez-Martín*; J. Martinez**; J.C. del Toro*; P. Núñez* & F. Sandoval*
*Dpto. de Tecnología Electrónica, ** Dpto. Lenguajes y Ciencias de la Computación
University of Málaga, Málaga (Spain)
{rvmartin,jmcruz,deltoro,pmnt,sandoval}@uma.es

Abstract: - The conceptual architecture of a robot is the organization of its actuation, perception and processing capabilities with the aim of generating a whole set of autonomous behaviours. In order to accomplish a task, autonomous mobile robots must be capable of perceiving its environment and maintaining an exact model of the world. Usually, a robot is equipped with several sensor systems to gather information from the environment and update this model. In this paper, we present a control architecture based on perceptions, which is specified from the sensors up to the highest level. The control architecture scheme follows the hybrid guidelines and maintains an environment representation where the highest level is built from perception outcomes. The system is composed of reactive and deliberative layers. In the reactive layer perceptions are organized into a set of modules divided into different levels of sensory representation: primitives, such as localization or landmark detection, and compound perceptions, such as feature maps (local SLAM). The other part of the reactive layer is the action module, where a set of behaviours provide the robot with navigation capabilities. Finally, the deliberative layer builds a symbolic environmental representation (topological map) and integrates suitable algorithms to accomplish the execution of a task.

Key Words: - autonomous robot, hybrid control architecture, active perception, framework, design patterns

1 Introduction

An autonomous robot can be defined as a machine entity that uses sensors to perceive its surroundings and acts in that environment by using its actuators. In order to achieve autonomous behaviour in an efficient and safe way, not only is it necessary to develop the appropriate capacities (perception, reasoning and action), but also to integrate them inside the robot in an efficient and robust way. Perception is especially significant if the environment is dynamic and unknown and only partial and imprecise information is available. For these systems, perception can be defined as the set of functions that obtain an abstract representation of the environment, directly usable by users or by the robot itself and the software that makes it autonomous. Therefore, perception and autonomy are tightly coupled: autonomy is possible thanks to the understanding, modelling and implementation of behaviours, based on a perception of the environment. Understanding of representations is built on top of these perceptions [8].

A robot control architecture can be defined as the overall organization of the different data processing modules in the robot and it controls the robot's interaction with the environment. The first robot control architectures were greatly influenced by classical artificial intelligence and they put the emphasis on symbolic representation of the environment and planning. These deliberative or hierarchical control architectures relied on the sense-model-plan-act paradigm (SMPA) and were unable to react in dynamic environments, because they depended heavily on the environment model. The intelligence in these control architectures originated from the designer and the robot itself had little or no autonomy. On the other hand, behaviour-based control [2] proposes the decomposition of the architecture into a collection of processes or rules that meet or accomplish several objectives. A behaviour-based approach considers intelligence to be demonstrated through meaningful and purposeful action in a given environment. Still, systems based on purely reactive behaviours, with little or no knowledge of the world, do not perform well when carrying out complex tasks. To achieve better performance, hybrid control architectures combine deliberative and reactive modules [1]. Thus, hybrid approaches try to make planning subordinate to reactivity and yet use it to guide reactivity at a high level. In order to bridge the gap between the deliberative and the reactive layers, these control architectures have a third layer, commonly called the

[1] This work has been partially granted by the Spanish Ministry of Education and Science: TIN2004-05961.

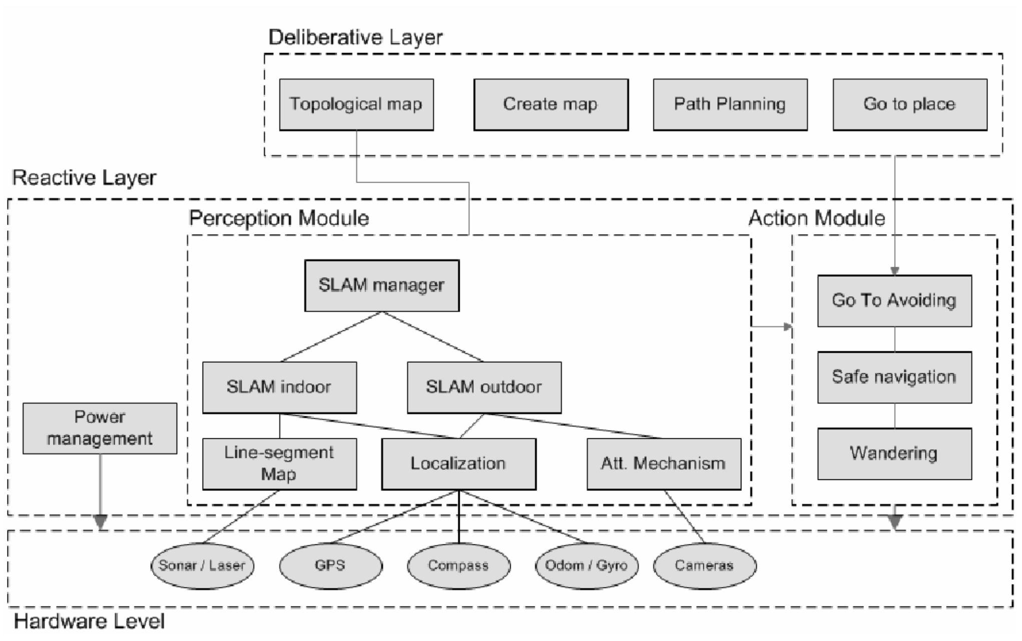

Fig. 1. Overview of the proposed architecture

task execution layer or sequencer. As has been previously mentioned, control architectures focused on behaviours or schemas [2] do not reach the desired level of autonomy. Besides, most of the hybrid architectures balance control toward the deliberative or the reactive layer.

This paper proposes a hybrid robot architecture that allows perceptual processes to control themselves and achieve more balanced control between reactive and deliberative layers thanks to the task execution module. In this approach the deliberative and the task execution layers are fused into one single layer. Thus, the architecture has been divided into two layers, one for perception generation and reactive control (reactive layer), and another one for deliberative algorithms and task execution (deliberative layer). The reactive layer is organised as a behaviour-based system where perceptions are hierarchically organised and they autonomously provide low-level and high-level information. Finally, the deliberative layer understands commands given by the user and controls the topological map generation. This paper has been organized as follows: Section 2 contains a brief description of related work. Section 3 presents the proposed hybrid architecture. Section 4 describes an object-oriented framework that has been developed to support the architecture. Finally, section 5 summarises conclusions and future work.

2 Related work

It is widely recognised that hybrid architectures are the most efficient ones, and there is general agreement on basic architectural principles [12]. Thus, the layout of the first implementation of a hybrid approach, the AuRA architecture [1], matches very well the three-layer decomposition of the typical hybrid architecture. AuRA was specifically designed for operation on a platform that carries out navigational tasks and has a hierarchical system for mission planning, a plan sequencer and a behaviour-based reactive layer. The mission planning layer is only activated when any contingency arises during the reactive layer execution. The XAVIER [15] and RHINO systems [3], developed later at CMU and Bonn University, respectively, follow the same guidelines.

Hybrid architectures that also correspond very well to the three layer decomposition are the 3T [5], the BERRA and the LAAS [6]. These architectures present well-defined deliberative, task execution and reactive layers. Perceptions are mainly grouped in the reactive layer. Therefore, they are directly controlled by the highest level of the architecture but not the other way around. That is, the response of perception modules is taken into account only when the deliberative layer decides.

It is worth noting that commercial platforms do not provide control architectures to achieve the desired level of autonomy [12]. For instance, Saphira is not organised in terms of layers. It represents the robot environment in a symbolic frame called Local Perceptual Space (LPS) and sends this information to each module in the architecture, regardless of their temporal or data abstraction level. Besides, there is no route planning, which means that the robot can only navigate to a point immediately accessible from the current position. Another example of a commercial architecture with no real deliberative layer is Teambots. Recently, ActivMedia Robotics has developed the ActivMedia Robotics Interface for Application (ARIA). This object-oriented software represents an important tool that takes advantage of its hardware abstraction and its capabilities to run cooperative behaviours.

In general, hardware abstraction is well handled in these architectures, but there are no interactive tasks, meaning there is only a low level of autonomy.

The architecture described in this paper is built over ARIA. In order to achieve the desired autonomy, we have to design new abstractions for high level concepts such as those required by reactive and deliberative layers. Thus, through ARIA, sensors supply their data to modules which extract higher order data. This information will be inserted into more elaborate perceptions by using an active perception strategy, which will be described in the next section. Therefore, the proposed architecture presents well-defined deliberative and reactive layers, which are in some way portable to other platforms.

3 Components of the architecture

The proposed system is based on perceptions, where stimuli are the key issue in the robot behaviour. In order to achieve autonomous behaviour in a dynamic environment, while the robot is carrying out a specific task, it must be capable of perceiving its environment and reacting to changes. There are different ways to implement the perception mechanism, based on the environment link, such as for example anchoring [14].

The architecture (see Fig. 1) is organized from the sensors up to the deliberative level. Perceptions are split into a set of modules that represent different stimulus and are organized in two levels: primitive and compound perceptions. The reactive layer is behaviour based, consisting of separate behaviours, where each one is designated a specific non-complex task in order to reach a target avoiding unexpected obstacles. The deliberative layer generates a symbolic environment representation (topological map), which is built from the perceptions outcome. On this layer, one part is dedicated to building this representation while the other two parts maintain the targets for accomplishing the task. Both layers have been developed following a new object-oriented framework in C++ that covers the functionality and interactions needed by each module. This framework is heavily based on common design patterns [4], and will be introduced in section 4.

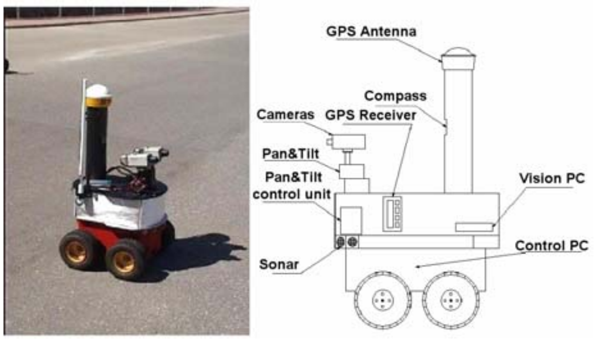

Fig. 2. Details of the mobile platform

Regarding hardware, our platform is based on a Pioneer AT from ActivMedia, equipped with two cameras mounted on a Direct Perception’s Pan and Tilt Unit (PTU), eight front sonar sensors, a GPS and a compass. It includes two embedded PCs, one of which is exclusively dedicated to vision purposes. Experiments have shown that this platform can work correctly in indoor environments [16] or outdoors [17][19]. Fig. 1 shows a schematic layout of the platform.

The following subsections describe the main characteristics of the layers in the proposed architecture. We focus on the details of the perception module, which has been designed to work in an autonomous and active way. From a functional point of view, the other part of the reactive layer (called action module in Fig. 1) and the deliberative layer are more similar to those present in other architectures, so they will be only briefly described.

3.1 Active perception module

To model perception as an active process means that perception is goal-directed and context-sensitive at every stage, including at the initial processing of input sensory data. Thus, active perception processes all data in a goal-directed manner and it greatly decreases the computational cost of perception because the system applies only specific computational resources to chosen parts of the sensory data. Therefore, active perception requires forming sensory plans from the information gathered by the robot from its environment, and transforms perception into a problem-solving process. This permits the robot to apply its learning capabilities to its perceptions. In order to perceive actively, the proposed architecture has access to lower-level sensory and motor information. Thus, it implements modules for landmark acquisition and localization and map building in indoor or outdoor environments.

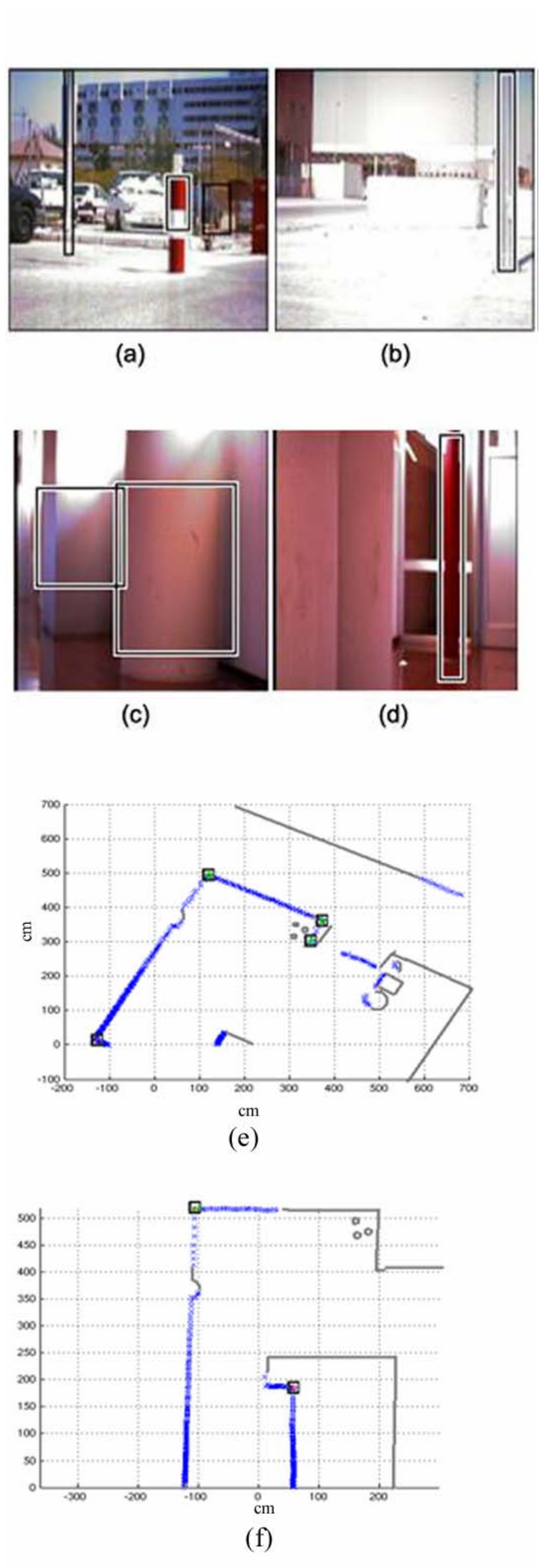

Fig. 3. Landmarks detected in different environments with: a-d) the visual attention mechanism, and e-f) line-segment/corner extraction (laser scan)

A. Autonomous landmark detection

The perception primitives (the lowest level of sensory representation) are composed of the localization module and two landmark detector modules. The line-segment map and the attention mechanism are based on landmark detection. These landmarks are the way to model the robot environment and are the key point for the feature maps built into the higher level of perception.

The type of landmarks depends on the sensor and the process used to acquire them. In [17] a visual landmark acquisition is presented. The proposed attention mechanism integrates bottom-up and top-down processing, selecting salient regions by computing different image features. Fig. 3a-d shows several examples of autonomous landmark acquisition. It should be noted that landmark detection is correctly achieved in indoor and outdoor environments.

Indoor environments present regular geometry (walls, corners, doors, etc). In order to take advantage of this information range bearing sensors can be used to extract these environmental features. Line-segment maps can be extracted from sonar [13] and laser scans [10]. Besides, due to the higher precision of the laser range-finder, it is possible to use a curvature based method to extract corners [10]. Figs. 3e-f present two scan data collected in an indoor environment. The laser scan range readings have been presented above the real layout.

B. Local Simultaneous Localization and Mapping

A mobile robot that has to embed autonomous motion abilities must necessarily be endowed with localization capabilities. In order to provide robust localization, an autonomous robot must integrate several methods, each of them fulfilling a particular requirement [8]. In our case, odometry computes positions thanks to the integration of the robot wheels' velocities. This position measure is prone to unbounded drifting, due to slippage and mechanical imprecision which provoke accumulative errors. These problems are even worse in outdoor environments, where the ground is not necessarily flat, but usually rough, and paths may be quite long. To obtain an accurate estimation of the robot pose, a GPS is used to provide an outdoor absolute position and a compass is used to refine the orientation pose [19]. GPS provides good accuracy in the position, but its availability depends on the environmental conditions (satellite signals can be blocked by buildings, trees, sources of electric and magnetic fields, etc) and it can only be used outdoors. In order to achieve an accurate and continuous estimation of the robot's position, a simultaneous localization and map building process (SLAM) [18] is used. Vision is employed to provide indoor and outdoor landmarks while sonar or laser sensors provide only indoor ones. The SLAM process is based on a features map. This map is built with the information provided by the corner-extraction/line-segment module or visual attention mechanism. The size of the feature map increases with the number of observations (2N+3). In large and/or dense environments these maps can reach a huge size, which means high storage and computational cost. In order to avoid these problems, a partial or local SLAM is used. Partial SLAM is based on removing from the state all the landmarks outside a local map around the current robot pose. The size of this local map is set based on the sensory horizon of the robot. Partial SLAM provides the robot pose and local landmark localization while the solution for the large-scale space representation and the closing of large loops (large-scale structural ambiguity) are postponed to the topological map.

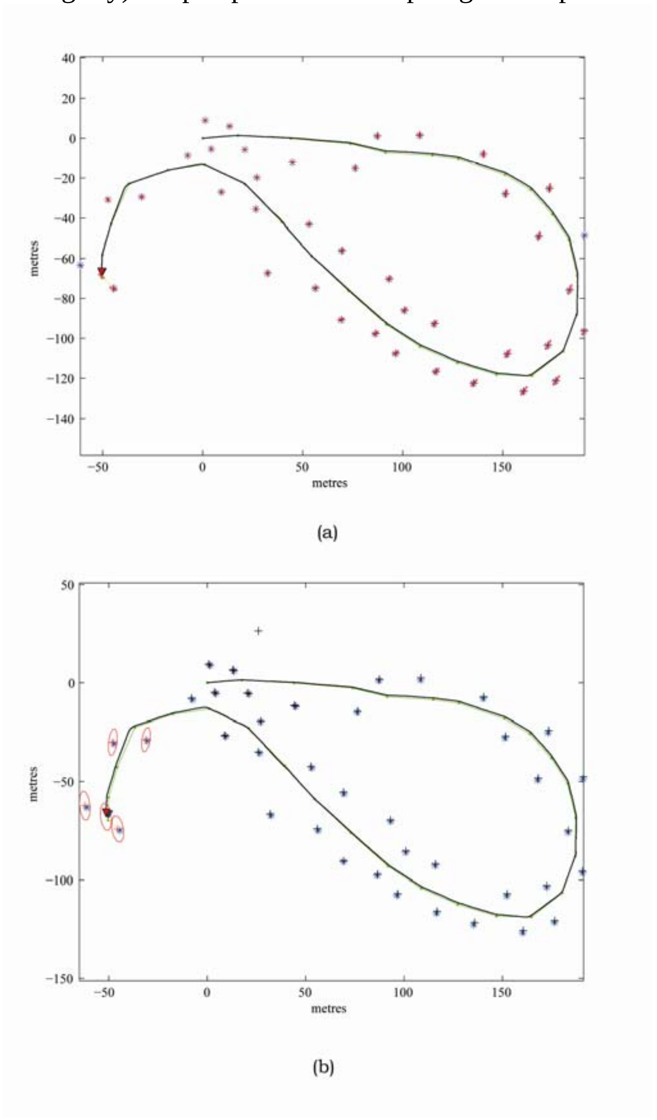

Fig. 4. a) Full and b) partial SLAM

Fig. 4 illustrates the difference between full SLAM (Fig. 4a) and partial SLAM (Fig. 4b). It can be observed that the uncertainty (red ellipses) in the vehicle pose and the landmark locations are higher in partial SLAM, due to the reduction when previously detected landmarks are observed again. In the case of partial SLAM this fact is less likely, but the uncertainty does not increase significantly.

Our approach implements two SLAM algorithms, one for indoor environments and another one for outdoors. The correct reception of GPS signals and the obstacle density are employed to detect indoor or outdoor environments. The SLAM manager shown in Fig. 1 decides which SLAM algorithm is more suitable to the robot environment.

3.2 Reactive navigation

As explained before, the reactive layer is split into several modules. The action module is implemented as behaviours and their outcomes are combined in order to select the desired action of the robot.

Firstly, an action called safe navigation stops the robot when the motors stall (there are no bumpers available to detect collisions). In order to navigate in dynamic environments, a behaviour that avoids obstacles is needed. Although different techniques can be used, the Obstacle Restriction Method has been implemented [9], because it obtains better results in dynamic environments. This behaviour maintains the goals that are passed by the deliberative layer and the wandering behaviour is used when there are no specific goals and the robot is exploring the environment.

3.3 Deliberative and task execution layer

Metric maps are not suitable for representing large-scale environments but provide a high degree of local accuracy and quantify uncertainty. On the other hand, topological maps provide a natural division of the environment, low computation and storage, large-scale connectivity and consistency. The strengths of topological and metric maps are complementary. Thus, many approaches that combine both paradigms exist [7]. These maps can be considered as topological representations based on information from local or global metric maps.

Local metrical mapping is based on the partial SLAM, which provides metric information on the robot's surroundings. The topological map describes the environment with a set of nodes (places) linked by edges (connecting paths). These nodes represent places that satisfy several conditions, for example areas with a high density of landmarks (SLAM) or interesting views (based on image features). The selection of these conditions is a key issue, due to

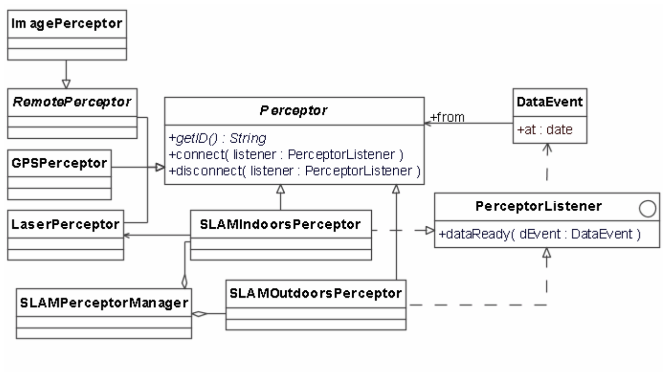

Fig. 5. Parts of the UML Perception module

the consistency of the map building process. At this level of representation the navigation problem is solved with a path-planning method to find a path in a graph, in this case the A* algorithm.

4 Implementation details

The architecture proposed in this paper has been implemented using a new object-oriented framework in C++. In the design of the framework we have taken into account all the high level abstractions needed by the reactive and deliberative layers. For instance, Fig. 5 depicts part of a UML class diagram [11] corresponding to the active perception module, which provides the required infrastructure to encapsulate and manage different sensor measurements. The abstract *Perceptor* class is the base class for all perceptors (primitive and compound) in the framework. It includes basic mechanisms to get data from the robot's sensors and to register *PerceptorListeners*. This are subscribers that will be warned when new data is available. Fig. 5 shows the relationship between *Perceptor*s and *PerceptorListener*s though a *DataEvent* object, which contains appropriate references to the caller and to the data available. Therefore, perceptions can be hierarchically organized, providing low-level and high-level information. Regarding specific perceptors, such as the wrappers for localization (GPS), line segment/corner extraction (laser), or SLAM, these are all singleton objects which have only one active instance in the system. The role of the SLAMPerceptorManager class also shown in the figure is that of deciding what the robot environment is, according to the results obtained by inspecting SLAMIndoors and SLAMOutdoors perceptor objects. Following the recommendations in our framework, complete use cases are built with few lines of code, as shown in fig. 6, where it is demonstrated how easily the hierarchical connection of perceptors is performed. First of all, the hardware-based LaserPerceptor and GPSPerceptor objects

```
int main(int argc, char *argv[])
{

  LaserPerceptor* lp = LaserPerceptor::getInstance();

 GPSPerceptor *gpsp = GPSPerceptor::getInstance();

 //registering the hierarchy: laser for indoors

  SLAMIndoorsPerceptor slamindoors;
  lp->connect(&slamindoors);

//registering the hierarchy: GPS for outdoors

  SLAMOutdoorsPerceptor slamoutdoors;
  gpsp->connect(&slamoutdoors);

//The manager will decide the appropriate environment

  SLAMPerceptorManager slam_manager;

  slamindoors.connect(&slam_manager);
  slamoutdoors.connect(&slam_manager);

  //Robot execution main loop

  Robot::getInstance()->run(argc, argv);

  return 0;
}
```

Fig. 6. A simple use case of the perception module

(wrappers for available laser and GPS) are connected to their corresponding SLAM algorithms. Note that these algorithms for outdoors and indoors locations may be possibly executing in separate threads (also decoupled of the robot execution main thread).

Regarding hardware-based perceptors, they are already available in the ARIA library. Our architecture provides the relationship among these primary (or basic) perceptors and a running Robot instance. For instance, code in fig. 7 shows the behaviour of the LaserPerceptor class. The connectToRobot method will be called from the Robot singleton instance in order to encapsulate the low-level procedures followed by ARIA. One of the most important tasks is to register a callback function to be executed when new laser data is available. Therefore, the newData method is responsible of notifying these data to those perceptors which were previously subscribed.

Another important concept in the framework is the possibility of having distributed perceptors. The *RemotePerceptor* class is prepared to act as a server

```
LaserPerceptor::LaserPerceptor()
{
  //wraps the ARIA laser
  laser = new ArSick();
  laser->configure(
            false,true,false,ArSick::BAUD38400,
            ArSick::DEGREES180,
            ArSick::INCREMENT_HALF);
}

void LaserPerceptor::connectToRobot(Robot *robot)
{
  functor1 = new ArFunctorC <LaserPerceptor>
                            (this,&LaserPerceptor::newData);
  laser->addDataCB(functor1,ArListPos::FIRST);
  robot->connector.setupLaser(laser);
  laser->runAsync();
  Logger::log("laser setup ok");

...

  if (laser->getDegrees()==ArSick::DEGREES180)
            Logger::log("degree Range: 180º");
  else
            Logger::log("degree Range: 100º");
  if (laser->getIncrement()==ArSick::INCREMENT_ONE)
            Logger:log("1º deg inc");
  else
            Logger::log("0.5 deg inc");
}

/*
The newData callback function is executed when new
data from laser is available in ARIA
*/
void LaserPerceptor::newData()
{

  std::list<ArSensorReading *>::const_iterator myit;

  static ArTime mylast;
  ArTime mynow;
  long int timeelapsed;

  mynow=laser->getLastReadingTime();
  timeelapsed=mylast.mSecSince(mynow);

  if (timeelapsed > 0){
    mylast=mynow;
    readings = laser->getRawReadings();
    DataEvent devent(this);
    sig_subscribers(devent);
  }
}

LaserPerceptor::~LaserPerceptor()
{
    laser->remDataCB(functor1);
    delete functor1;
    delete laser;
}
```

Fig. 7. The LaserPerceptor behaviour

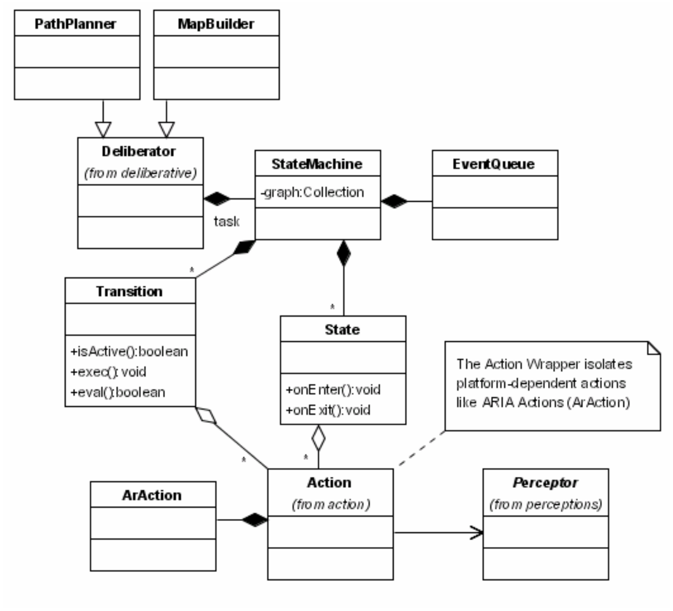

Fig. 8. Class diagram of *Deliberators* and *Actions*

that gets network data using TCP/IP sockets. In our mobile platform, it is very useful to offer load balancing among different PCs. With our approach (see Fig. 1) one processor is responsible for computing image inputs from cameras, a resource-expensive task, whereas the other processor contains the deliberative and reactive layers built over ARIA. Therefore, the image data from cameras is reconverted to *regions of interest*, which are sent to a specific *ImagePerceptor* derived from a *RemotePerceptor*, giving objects in the architecture the possibility of accessing those data with the minimum delay (by using UDP datagram sockets).

Although this paper has focused on the perception part of the control architecture, we are now implementing the functionality requested by the deliberative layer. Fig. 8 shows the features available in the framework for organizing high level tasks through so-called *Deliberator* objects. Their default implementation contains state machines to represent the behaviour corresponding to a deliberative task. *State*, *Transition* and an *EventQueue* are classes which compose each state machine. *Transition* (and *State*) classes may contain ordered sequences of *Action* objects, which are executed only when they are active, that is, when a specific *DataEvent* has been received. Regarding *Action* classes, they implement the command pattern, acting also as a *wrapper* for platform-dependent actions, like the ones available within the ARIA platform, as shown in Fig. 6.

The design of this framework provides clear extension options for developers in our group so that they may provide their own implementations of new *Perceptors*, *Actions* and *Deliberators* as needed.

5 Conclusions and future work

This paper describes a hybrid architecture for a mobile robot that allows perceptual processes to control themselves and achieve more balanced control between reactive and deliberative layers. We have introduced its main components focusing on the active perception module, which has a critical role with respect to robot operation. It provides a hierarchy of autonomous perceptions that convert low-level information into high-level data. We have also presented a framework for implementing the functionality needed by reactive and deliberative layers. The aim of this framework is not only to reduce the development time but also to provide platform-independent high level abstractions of the components in the architecture. Work in the immediate future consists of obtaining new Perceptors for the vision mechanism, in order to recognize nodes in a topological map. This will constitute the basic support for an ongoing *path-planning* task, which will improve the implemented algorithm.

References:

[1] Arkin, R. Behaviour based robotics, MIT Press, 1998.

[2] Brooks, R. A robust layered control system for a mobile robot. IEEE Journal of Robotics and Automation, Vol. RA-2, No. 1, 1986, pp.14-23.

[3] J. Buhmann, W. Burgard, A.B. Cremers, D. Fox, T. Hofmann, F. Schneider, J. Strikos, & S. Thrun. The mobile robot RHINO. AI Magazine, Vol. 16, No. 2, 1995, pp. 31–38.

[4] Gamma, E.; Helm, H.; Johnson, R. & Vlissides, J. Design Patterns. Addison-Wesley, 1995.

[5] Gat, E. On Three-Layer Architectures. In Artificial Intelligence and Mobile Robots, MIT/AAAI Press, 1997, pp. 195-210.

[6] Ingrand, F. & Py, F. An Execution Control System for Autonomous Robots. Proc. of the IEEE International Conference on Robotics and Automation, 2002, pp. 1333-1338.

[7] Kuipers, B.; Modayil, J.; Beeson, P.; MacMahon, M. & Savelli, F. Local Metrical and Global Topological Maps in the Hybrid Spatial Semantic Hierarchy. Proc. of IEEE Int. Conf. on Robotics and Automation, 2004, pp. 4845-4851.

[8] Mallet, A. Localisation d'un robot mobile autonome en environnements naturels. PhD thesis, Institut National Polytechnique de Toulouse, July 2001.

[9] Minguez, J. The Obstacle-Restriction Method (ORM) for Robot Obstacle Avoidance in Difficult Environments. Proc. of the IEEE Int. Conf. on Intelligent Robots and Systems, 2005, pp.3706-3712.

[10] Nuñez, P., Vázquez-Martín, R., Bandera, A., Sandoval, F. Feature Extraction from Laser Scan Data Based on Curvature Estimation for Mobile Robotics. IEEE Conference on Robotics and Automation, 2006. pp. 1167-1172.

[11] Object Management Group. Unified Modelling Language Specification, 1.5. Available at http://www.omg.org, Accesed: 2005-07-07

[12] Orebäck, A. & Christensen, H. I. Evaluation of architectures for mobile robotics. Autonomous Robots, 14, 2003, pp. 33-49.

[13] Pérez-Lorenzo, J.M.; Vázquez-Martín, R., Núñez, P.; Pérez, E.J. & Sandoval, F. A Hough-based method for concurrent mapping and localization in indoor environments. Proc. of the IEEE Conf. on Robotics, Automation and Mechatronics, 2, 2004, pp. 840-845.

[14] Saffiotti, A. & LeBlanc, K. Active perceptual anchoring of robot behaviour in a dynamic environment. Proc. of the IEEE Int. Conf. on Robotics and Automation, 200, pp. 3796-3802.

[15] Simmons, R. G. (1994), Structured Control for Autonomous Robots. IEEE Trans. on Robotics and Automation, Vol. 10, No. 1, 1994, pp. 34-43.

[16] Urdiales, C.; Pérez, E.J.; Sandoval, F. & Vázquez-Salceda, J. A hybrid architecture for autonomous navigation using a CBR reactive layer, Proc. of the IEEE/WIC Int. Conf. on Intelligent Agent Technology, 2003, pp. 225-232

[17] Vázquez-Martín, R.; del Toro, J.C.; Bandera, A. & Sandoval, F. Data and model-driven attention mechanism for autonomous visual landmark acquisition, Proc. of the IEEE Int. Conf. on Robotics and Automation, 2005, pp. 3383-3388.

[18] Vázquez-Martín, R., Nuñez, P., del Toro, J.C, Bandera, A., Sandoval, F. Adaptive Observation Covariance for EKF-SLAM in Indoor Environments using Laser Data. 13th IEEE Mediterranean Electrotechnical Conference, 2006, pp. 445-448.

[19] Vázquez-Martín, R.; Pérez, E.J.; Urdiales, C.; del Toro, J.C. & Sandoval, F. Hybrid navigation guidance for intelligent mobiles. Proc. of the IEEE Int. Conf. on Vehicular Technology, 2006.