A flexible platform for vision-based robot navigation

Enric Celaya, José Luis Albarral, and Tom Creemers

Instituto de Robótica e Informática Industrial (UPC-CSIC)

Issues in Robot Navigation

The most important capability for a mobile robot is *navigation*.

Navigation is not a goal by itself, but a necessary requirement to perform tasks in which the robot has to move towards the operation place.

Many approaches to robot navigation define the navigation task as a goal by itself, detached from its intended use in practical applications.

Their main emphasis is in precise localization, path planning, and map building.

However, little attention has been given to the role of the user in aspects such as *target definition* or *tele-operation* in the navigation task.

Navigation tasks and environments

Many works assume repetitive navigation tasks, most often in indoor environments.

Repetitive tasks:

- Commonly performed indoors, where navigation can take profit of the structure of the environment (flat floor, perpendicular walls, doors, constant illumination,...)
- A map is often available and, if not, it is worth spending some time letting the robot wander through the environment to build one by itself.
- The goal position is defined as a location on the map, also available to the user.

Non-repetitive tasks:

- Often in outdoor environments, no special structures can be expected.
- Maps rarely available.
- The environment may be unknown to the user, as well as to the robot, so that even the definition of the navigation task involves some kind of exploration.

Vision-based navigation

Using vision is the most natural way for a user to define a navigation target in previously unknown environments.

Vision-based navigation uses the visual identification of the target and some selected salient landmarks to guide the robot towards its goal.

The visual definition of the target involves some inaccuracy in its absolute position.

Remark

Robot localization defined with respect to:

- the start position -> error increases with robot movement.
- the goal position -> gets more accurate as navigation proceeds.

> In vision-based navigation, accurate robot localization becomes less important than reliable goal and landmarks identification.

Requirements of a user interface

For the definition and monitoring of vision-based, non-repetitive navigation tasks in non-structured environments, we need:

- A visual interface to define the desired target and monitor robot progress.
- An autonomous navigation process that can drive the robot to its target.
- The possibility to take direct control of the robot (teleoperation).
- Automatic self-protecting reactions of the robot when the user fails to avoid a danger.

A platform for robot navigation

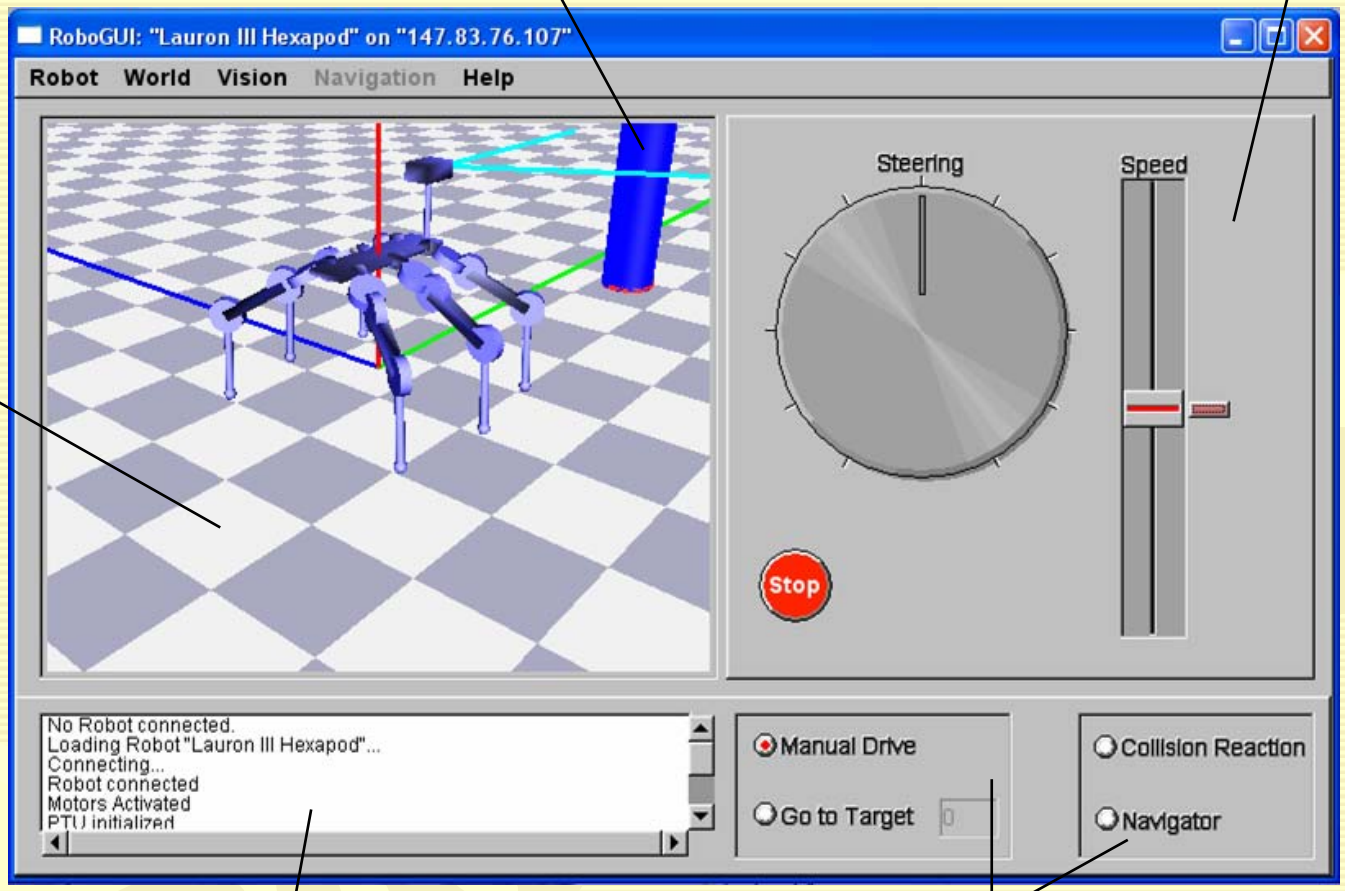
A flexible vision-based navigation control platform, not specific of a particular robot, has been developed.

The functionalities provided by the interface are defined at an abstraction level valid for virtually any kind of mobile robot.

The expected capabilities of the robot for its use with the interface are:

- Moving in a commanded direction (following a circular trajectory of given radius)
- One or more cameras, with pan & tilt capability.
- Sensors for obstacle detection (proximity or contact).
- Some form of odometry for short travels.

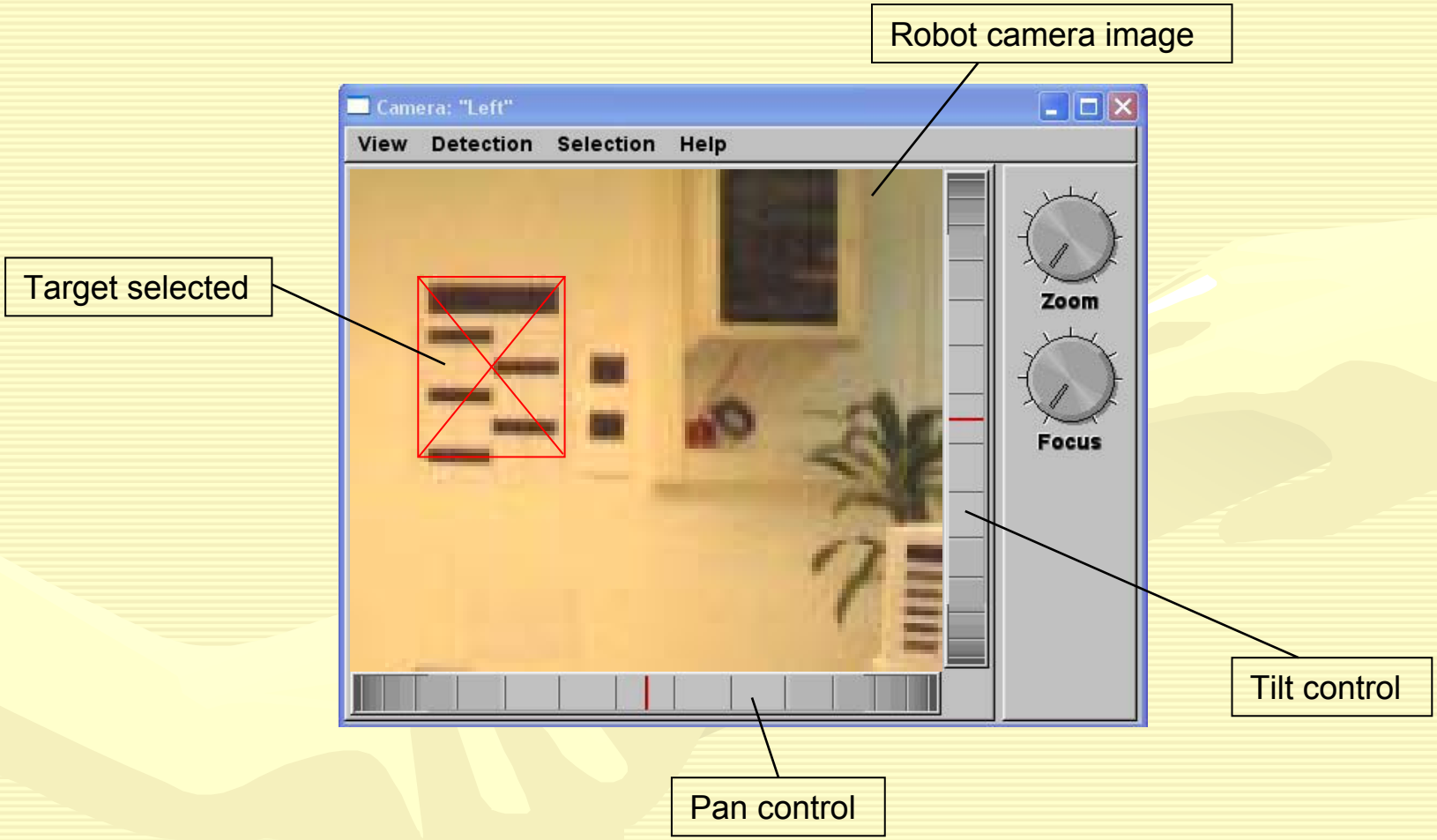
The user interface



Target selection

- A *camera view* window can be open for each robot's camera
- Manual pan & tilt control, independent for each camera, is available.
- A *landmark detection* algorithm can be selected: found landmarks are displayed on the image.
- The user can select a detected landmark as camera target to get an automatic visual tracking of the landmark with this camera.
- A landmark can also be selected as the navigation target for the robot.

Target selection



Manual robot control

The instantaneous motion of the robot is determined by its linear and angular speeds: v, ω.

The robot will describe arcs of circumference of radius R = v/ω.

The user can control v and ω by means of the speed cursor and the driving wheel.

To achieve coherent control including turn in place, the following relationships are used:

$$\omega = c \sin(\alpha) / L$$

$$v = c (1-\sin(\alpha)),$$

where:

- c is the speed position set by the user
- α is the angle of the driving wheel
- L is a constant that determines the influence of the steering angle in the turning radius, and intuitively, must be related to the size of the robot.

Pilot control

Even if the user chooses to drive the robot manually, he can find too hard to follow a trajectory that does not collide with any obstacle.

To facilitate this task, the interface provides a *pilot* capability that modifies the local trajectory when an obstacle is detected right in the path of the robot. The pilot can also provide a lower protection level when a navigation algorithm is used.

- The pilot makes no use of information about the target position or obstacles stored in an internal map (though such information may be used by the *navigator*).
- The pilot only reacts to obstacles currently detected by sensors, and that higher level processes (user control or the navigator) were not able to avoid.
- Obstacles detected in the path that would cause a collision can be avoided by a robot turn.
- Actual collisions and other blocking situations produce a reaction consisting in a short movement in reverse direction, a turn in place, and proceeding with the initial velocity.

Simple autonomous navigation

The *go to target* option can be used to drive the robot towards the goal in easy environments.

Its working depends on the visual tracking of the target: the angle of driving wheel is set to the perceived direction of the target.

Its simultaneous use of the pilot allows it to get ride of many obstacles that can be avoided without the need of a path planning for alternative routes.

The *go to target* facility can not be considered as a real navigation algorithm: it does not use a map and does no path planning. It is just an automatic attractive reaction to the currently seen target.

The fact that such a simple strategy works in so much situations imposes a minimal efficiency level to the expected performance of a real, more complex navigation algorithm.

Control with a navigation algorithm

Once the user has selected a navigation target, he can pass the control to an autonomous process governed by a navigation algorithm.

Several different navigation algorithms may coexist in the system, so that the user can choose the preferred or more appropriate one for the task at hand.

Normally, the pilot will remain active during its operation to grant save movement despite possible flaws in the navigation algorithm.

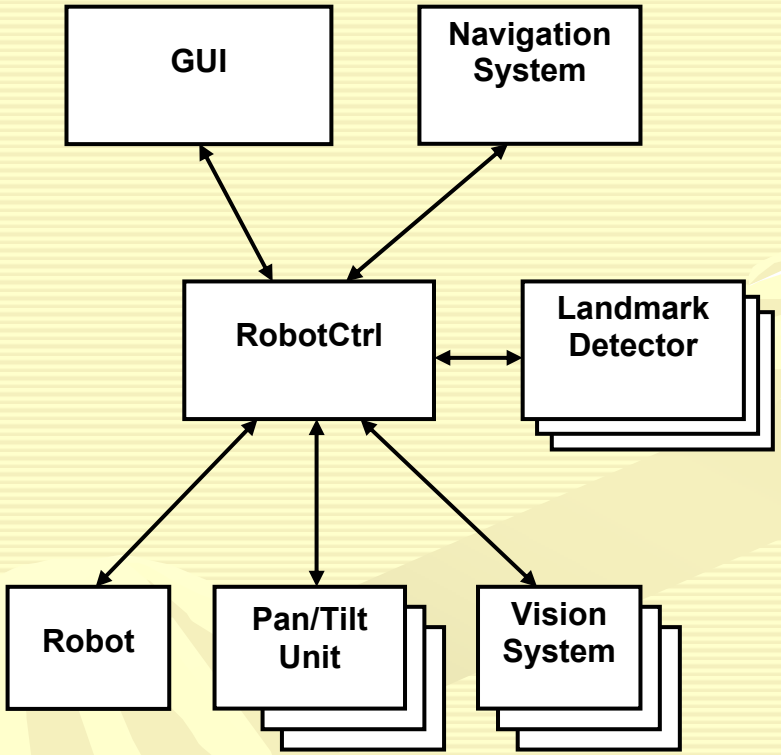
The user can interrupt the navigation process at any time by taking manual control, and resume the algorithm execution when desired.

Also, the user can define a new target without stopping the navigation process, so that the algorithm will forget the old target to follow the new one.

Implementation

- The platform has been designed as a collection of modules that communicate in pre-established ways.
- The core of the system is the RobotCtrl module, which is under direct management of the GUI interface and coordinates the operation of the other modules.
- All the other modules can be re-implemented independently, and combined to constitute the whole system.
- There are hardware-dependent and hardware-independent modules.
- Hardware-dependent modules must be implemented once for each different hardware element that will be used.
- Hardware-independent modules can have multiple implementations that will be selectable by the user, and are compatible with all implemented robots.

Implementation



Hardware-dependent modules

Robot module:

Translates the abstract commands for robot motion and sensor acquisition used in the interface to the corresponding commands of the specific robot. The required sensorial information includes collision detection and odometry.

Pan/tilt Unit module:

To point with the cameras attached to each pan&tilt unit in the desired direction.

Vision System Module:

Must provide, for each camera, the functionalities of image acquisition and, if available, the control of zoom, focus,...

Hardware-independent modules

Landmark Detector module:

It receives a camera image and returns a list of detected landmarks with unique identification labels and relative position estimation.

Navigation System module:

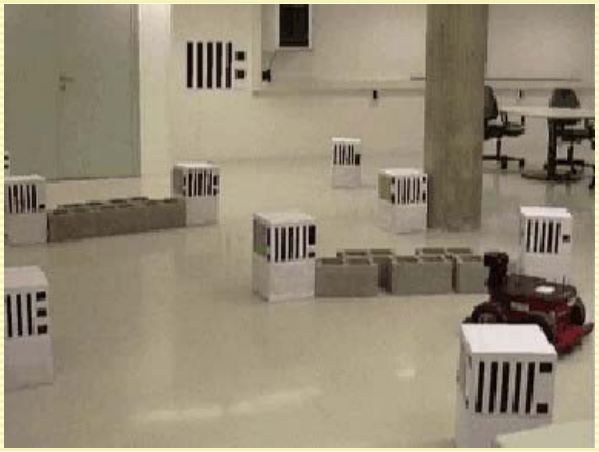
Receives the list of detected landmarks, the target identification, obstacles information and odometry measurements, and returns a driving command for the robot.

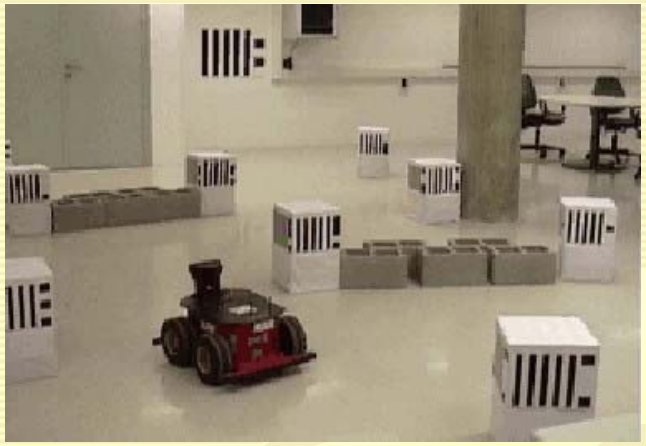
Possible extensions of the system may include independent modules for Visual Tracking and Pilot.

Demonstration example

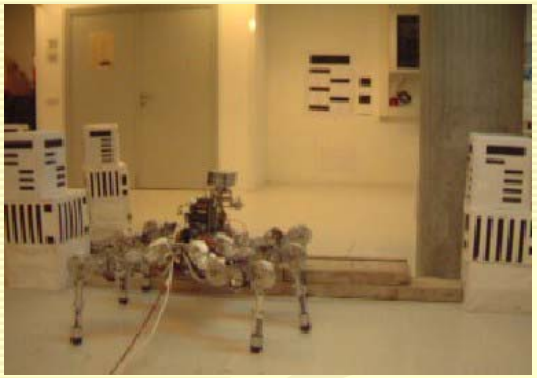
- The system has been tested with two different robots, a wheeled Pioner II from RWI and a sisx legged Lauron III from FZI.
- When confronted with the same environment, and using the same navigation algorithm, the resulting path is qualitatively different in both cases.
- The wheeled robot detects an obstacle that can not be surmounted, and the navigation algorithm plans a path around it.
- The legged robot detects the same obstacle but it can surmount it without informing the navigation algorithm of it.

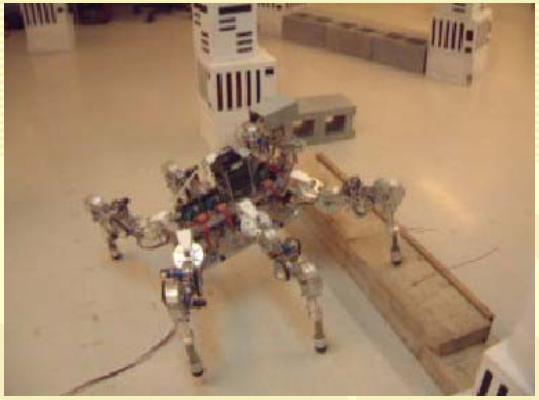
Navigation with a wheeled robot





Navigation with a legged robot





Conclusions

- Different mobile robots can be controlled with identical interface
- The user can ignore most aspects of the particular robot
- Modular design allows interchanging algorithms and robots
- The platform provides a tool to perform comparisons between algorithms, robots or environments.

Acknowledgements:

This work has been partially supported by the Spanish Ministerio de Ciencia y Tecnología and FEDER funds, under the project DPI2003-05193-C02-01 of the Plan Nacional de I+D+I.