A Navigation Agent for Mobile Manipulators

Mario Haut[1], Luis Manso[1], Daniel Gallego[1], Mercedes Paoletti[1], Pablo Bustos[1], Antonio Bandera[2], and Adrián Romero-Garcés[2]

[1] University of Extremadura, Cáceres 10003, Spain
pbustos@unex.es http://robolab.unex.es
[2] University of Málaga, ISIS Group
Málaga, Spain

Abstract. Robot navigation and manipulation in partially known indoor environments is usually organized as two complementary activities, local displacement control and global path planning. Both activities have to be connected across different space and time scales in order to obtain a smooth and responsive system that follows the path and adapts to the unforeseen situations imposed by the real world. There is not a clear consensus in how to do this and some important problems are still open. In this paper we present the first steps towards a new navigation agent controlling both the robot's base and the arm. We address several of theses problems in the design of this agent, including robust localization integrating several information sources, incremental learning of free navigation and manipulation space, hand visual servoing in camera space to reduce backslash and calibration errors, and internal path representation as an elastic band that is projected to the real world through measurements of the sensors. A set of experiments are presented with the robot Ursus in real and simulated scenarios showing some encouraging results. . . .

Keywords: robotics, navigation, social robots

1 Introduction

Robot indoor navigation and manipulation are crucial components of current autonomous robot control architectures. As other complex modules that form part of these architectures, navigation and manipulation are usually decomposed in several loosely coupled elements that form a distributed system. Typical navigation elements are collision avoidance, environment perception, (re)planning of safe optimal paths and (re)localization. One challenge in the design of these architectures is the "gap" problem, that arises when two different elements have to share enough context as to take proper informed decisions. A typical example would be the gap between the local collision controller and the path planner. Another one is the gap between the (re)localization algorithm and the environment perception element, when a moving human crosses in front of the robot, or when a new structural element appears in the environment.

Different solutions have been proposed to this problem, mainly to specific versions of it. One of the best known ideas is the concept of elastic bands, introduced by [1]. An elastic band is a theoretical construction obtained by a path planner that gets grounded to the real world by means of the interaction with a range sensor. The band works as a glue filling the gap between the internal representation of the path and the constraints imposed by the world physics. The path could be "broken" by a human passing by and restored afterwards. The local controller only "sees" a small perturbation that might involve a change in speed. In this paper we present ongoing work on the design of a new navigation and manipulation agent for the RoboCog architecture, that is based on the idea of elastic bands. Figure 1 shows a schematic representation of the robot in a navigation task that ends with the manipulation of the cup on the table.

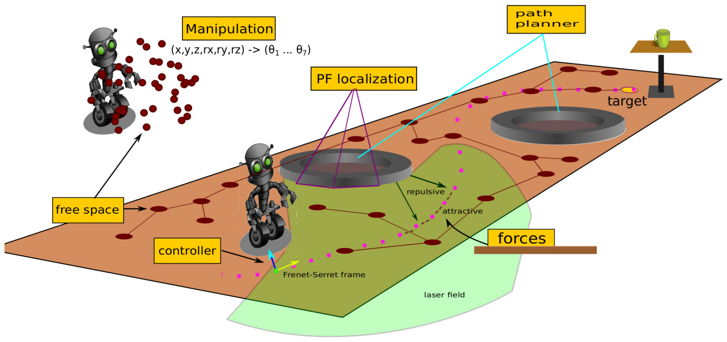


Fig. 1. Schematic view of a robot in a BringMe(x) task showing representing the free space for base navigation and manipulation. Dotted line is the planned path partially smoothed by laser projecting forces. Knowlegde of obstacles provides localization information.

Following these ideas, we will address several problems in the design of a new manipulation and navigation agent that will replace our current technology. We will describe improvements for indoor localization, lifelong free space learning and manipulation control with poorly calibrated and backslash mechanical arms. These three issues are crucial to achieve the mid-term goal of having a reliable BringMe(X) skill in the robot. There other elements necessary to accomplish this endeavor but they are discussed elsewhere [2] [3]. The algorithms described here are based on a shared representation of the robot and the environment. The different software components can access this structure to share and become aware of what other modules are doing. We review now work done in the elastic band concept and in visual servoing, since those are the main theoretical motivations in this paper.

Elastics bands were introduced by Quinlan and Kathib [1] as a method to close the gap between path planning and real-time collision avoidance. The technique has not received too much attention in the robotics community, maybe due to the independent interest on more specialized methods of local control and path planning. Later on an extension of the idea was introduced as elastic strips to cope with robots with many degrees of freedom[4]. All DOFs of the manipulator where included combining obstacle avoidance with desired posture behavior. A last generalization to the original idea was published as elastic roadmaps [5] to include planning in the loop.

When there are unmodeled misalignments and backslash in a low cost robotic arm, the place where the hand will get and the place that the camera is selecting, will not match. One elegant way to solve this problem was described by Hutchinson [6]. The idea is to bring the hand and the target to the camera space and perform a visual control loop there minimizing the observed visual error. The process can be shown to converge under mild conditions. The theory comes form visual servoing, a mature discipline originated in the control arena that has spread many new research lines and applications. The original work of [7] was followed by many others [8] [9] [24] [10] and also in different areas such as tracking of unknown objects [11] or binocular heads [12].

The rest of the paper describes the overall system and each of the problems and improvements introduced in the architecture. A final section discusses two experiments involving the robot Ursus that validate the choices made here and the way they have been implemented and integrated.

2 Overview of the System

The navigation agent proposed here is being built as part of the robotics cognitive architecture RoboCog [13] [2][3]. The overall goal is to integrate navigation and manipulation in a common framework so more complex tasks can be handled through body, head and arm coordinated movements. In this paper we will focus on the first steps of the design of the agent, describing localization, learning of free space, path planning, path execution and trajectory control of the arm. Each part is described as an improvement over their previous versions in RoboCog. We have identified important limitations in each one and introduced the necessary changes to solve the existing problems. From the point of view of the software, the complete agent is being built using the robotics framework RoboComp [14]. Each functionality is implemented as a set of interconnected components, many of which are shared among the others.

2.1 Internal Representation of Space

In RoboCog, the robot and its environment is represented as a graph $I = (N, E)$, also called *InnerModel*. The nodes in I correspond to parts of the robot and to elements in the world. They belong to one of the following types $N \rightarrow \{Robot, Joint, Laser, IMU, RGBD, Object, Mesh\}$. The edges in I are rigid euclidean transformations linking the nodes and represented as $4x4$ homogeneous matrices. The nodes can be extended with a list of $< attribute, value >$ pairs so they can be annotated with semantic information encountered during robot operation. In building the new Navigation agent, we have extended the initial set of types with a new one, named FreeSpace. This type defines a graph $G = (N, E)$ representing what the robot believes about its free configuration space. We now define separately the C-space of the robot base and the arm. Let's as $B \in \mathbf{R^2}$ be the base's C-space in which orientation is ignored, and $A \in \mathbf{R^7}$ the C-space of the arm. From them we can define two free configuration spaces, $B_f = B/C_{obs}$ for the displacement of the base, ignoring orientation, and $A_f = A/C_{obs}$ for the movement of the arm. Therefore, two graphs will be created, $G_B = (N_B, E_B)$ and $G_A = (N_A, E_A)$. In the next sections, the construction of these spaces and their use in computing safe paths will be described.

2.2 Localization

Localization is a crucial task that updates the estimated position of the robot in its internal model. Probabilistic algorithms based on partially known geometric information of the environment constitute the focus of this task. We use the recently introduced particle filter variant CGR [15] to obtain an an estimate of the pose of the robot with respect to an initial global reference system. The map of the environment is represented as a list of lines corresponding to the lower part of walls and known objects in the world. This algorithm is very efficient in cpu cycles and number of particles because the measurement function is analytic and derivable. A pre-ordering of the lines accounts for occlusions and an internal minimization loop improves the particles position using the Jacobian of the measurement. To obtain a reasonable estimate of the ground truth pose of the robot, a set of AprilTags [16] marks have been distributed in the apartment. Specialized components detect the marks and compute the error between the robot's current belief and its real position.

Figure 2 shows the current organization of components that implement localization. Arrows in the graph can be interpreted as one component sending information to another. *Localizer* receives poses from Base, AprilLoc and CGR along with their uncertainty, although only the base odometry and CGR estimation are fused. Localization based on AprilTags are used only for evaluation. Currently, an empirically obtained threshold over the variance is used to combine them and produce the current belief for the robot.

2.3 Free space representation for the robot

If we want the robot to be perceived by the human as real collaborative object, it has to react and operate at human rates. One limitation to this requirement is the delay introduced by path planning algorithms searching the free space, such as RRT [17]. For our agent we use the probabilistic road map algorithm, PRM [23] to create a graph representing the free space, and RRT only to search for paths when unconnected islands remain in the PRM graph. The resulting path is inserted into the graph to connect the isolated islands. RoboComp currently includes a wrapper for OMPL [19] but, as of today, OMPL's PRM implementations does not allow to store the computed graph on disk. As that feature is

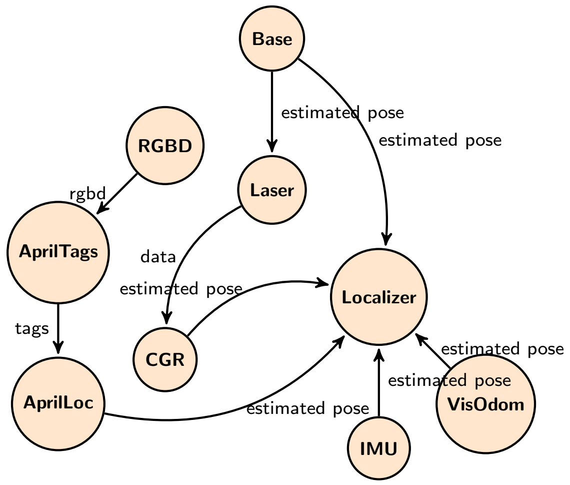


Fig. 2. Components used for localization in the Navigation Agent: Base controls the robot displacement, RGBD provides access to the camera, Laser provides access to the laser, AprilTags detects and estimates the pose of the tags, AprilLoc computes the localization error between the current estimate and what the tag provides, CGR implements the Corrective Gradient Refinement algorithm, IMU and VisOdom estimate indepent robot poses but their ouputs are not included here, and Localizer is the component that integrates all the sources to maintain the current belief.

crucial for long term robot operation we implemented a version of PRM using the Boost Graph Library, BGL.

A free space graph created with our PRM algorithm is shown in Figure 3. The upper-right rooms has been intentionally placed there to create an isolated region in the graph, giving a finite construction time. In (b) the algorithm calls RRT and obtains a feasible path. The robot traverses the path in (c) and in (d) it is added to the graph a only one connected component remains.

2.4 Free space representation for the robot's arm

A similar argument concerning the planning time can be applied to arm manipulation, where seven DOFs expand the configuration space. Natural HRI demands that repetitive actions improve over time. If the robot grabs a cup from the same table several times, the accompanying person will expect that the robot reduces its execution time down to human standards. Doing otherwise, human confidence on a helpful interaction decreases. To avoid this situation we adopt a similar solution as in the previous section. In this case, instead of a randomly sampled graph of free C-space, a regular 3D grid sampling the working volume of the arm will be used. Each element in the grid codes an euclidean 6D pose for the hand tip and a set of configurations in free C-space that correspond to that pose. In this initial model, only the most common hand orientation for grasping a common small object is assigned to each point in space. As we will see later, this

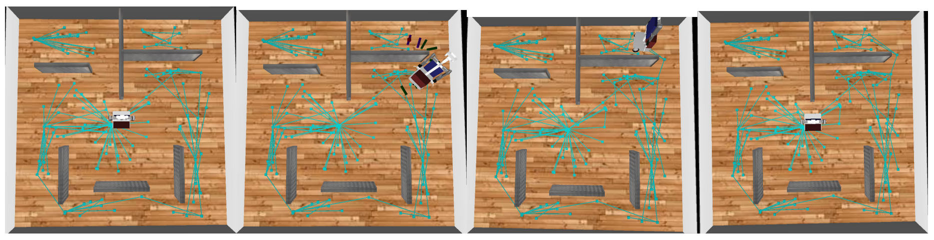

(a) Step I (b) Step II (c) Step III (d) Step IV

Fig. 3. Sequence showing graph learning with PRM in N_B created by PRM. (a) The upper right room has a very narrow entrance to force the creation of an island. (b) When the robot is sent there, RRT is activated and finds a route to the isolated spot. (c) The robot traverses the path and (d) the steps in the path are learned into the graph. Only one connected component remains

is not a limitation since the final manipulation plan includes a last refinement step, using inverse kinematics, to reach the target orientation and position.

2.5 Path planning in free space

Once these graphs have been created using inverse kinematics computation, the process to obtain a safe path is quite similar in both robot navigation and arm positioning. Note that both graphs are currently kept separated although they will be integrated in a near future. The path is created by first searching the closest point in the graph to the current position -robot or hand-, then the closest point in the graph to the target position and, finally, a path through the graph linking both points.

The plan constructed by the path planner can be seen as a *theoretical* construct based on the robot's beliefs about the world. As such, it will not be exact or even precise. Therefore, another components are needed to *ground* this construct into the real word using the information coming for the sensors. These new components update the path as it is traversed, adapting it to unexpected events and detect critical conditions blocking the robot's path. So, as the path is a shared structure, when a blocking situations occurs, the planning agent is aware and reacts computing a new path. This dynamic process will continue until the path is completed or no solution can be found in a certain amount of time. Note that the path connects the global planner with the robot controller providing the necessary persistence to avoid local minima.

As illustrated in Figure 1, the path is *analyzed* under the laser field and two virtual forces are created that push the path away from the obstacles while keeping it from bending too much [1]. Currently we use the following robot controller to compute the final forward speed V_a and rotation speed V_r [20]:

$$\begin{aligned} V_a &= V_M e^{\alpha C} G(\beta V_r) \\ V_r &= \phi + \arctan(d) + C \end{aligned} \tag{1}$$

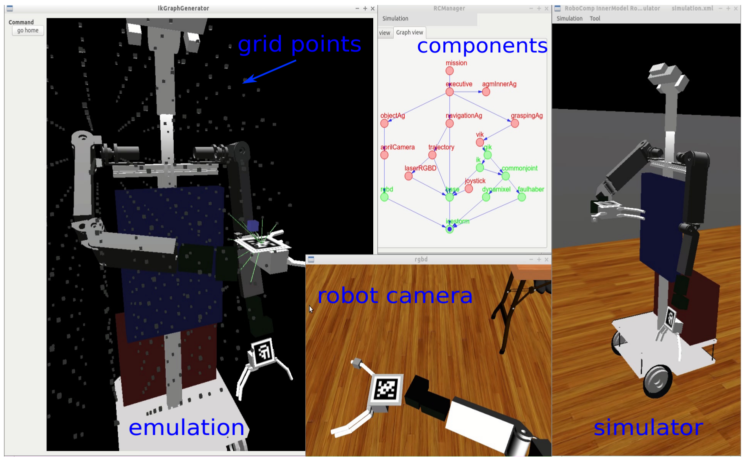


Fig. 4. Free space graph for the arm. The four captures express different aspects of the process. On the left, the robot is shown with grid during the learning process - emulation. Middle up, is the graph of components deployed for this activitiy. Middle down, is the arm as seen by the robot's camera. Left, is the robot in RoboComp simulator.

where V_M is the maximum advance speed, C is the curvature of the path, G is the Gaussian function, V_r is the rotation speed and β is a gain. Note that V_a is computed as the inhibition of the initial V_M caused by world interactions. For V_r, ϕ is the angle formed by the robot and the tangent to the path at the closest point to the robot, d is the perpendicular distance to the road and C is again the curvature.

The dynamics created by the algorithm make the path adapt to narrow passages, moving objects and even can be broken by a person passing by and restored afterwards. As the robot traverses the path, the steps left behind are deleted and the process ends when the target is reached and the path is completely erased.

2.6 Path execution by the arm

Low-cost arms present an important limitation that must be solved. Backslash and inaccurate calibrations induce errors in the end effector's position. The errors can easily ruin any intent of grasping objects if the only feedback used is the one from the joints' encoders. An elegant way out of this situation is to use a two-step procedure. First, the hand is brought to a place close to the target using a path through N_A. This movement is based only on joints feedback and is executed in parallel with a movement of the eyes whose purpose is to bring the hand and the target inside the camera's frustum. The planning of that movement using the graph of free space N_A was described in Section 2.5. The second step is a closed-loop visual servo control that brings the hand to its final grasping positions. This second part is the one that can cancel out the errors caused by

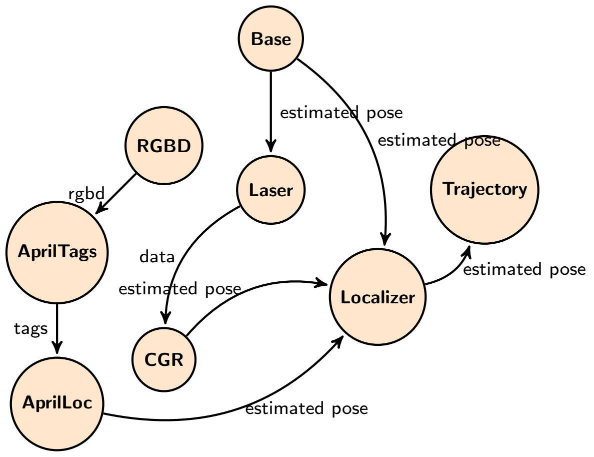


Fig. 5. Components involved in free space learning, path generation and navigation. Note that most components are also used in graphs shown in other sections.

backslash and mechanical misalignments. To detect and track the position of the hand, an AprilTag [16] is used. We present now this algorithm.

To describe the scenario and the operations involved we follow the notation in [6]:

- ${}^{x}R_{y}$: expresses the rotation matrix of the coordinate frame y with respect the x frame
- ${}^{x}t_{y}$: the position of the frame y with respect x
- ${}^{x}t_{y}$: the position of the frame y with respect x

We also define the following poses[3]:

- v_i: The coordinate frame of the end effector, as provided by the visual feedback in the ith iteration of the algorithm.
- t: The target pose.
- k_i: The pose that is actually sent to the IK module in the ith iteration of the algorithm. The existence of backslash and calibration inaccuracies make t and k differ.

Thus, the aim of the algorithm is to reduce to the maximum the difference between the effector's pose as given by visual feedback (v) and the target pose (t). Following the previously mentioned notation, the translation error is computed as ${}^{t}t_{v}$ and the rotational error as ${}^{t}R_{v}$. This way, the goal is to make ${}^{t}t_{v}$d as close to zero as possible and ${}^{t}R_{v}$ as close to an identity matrix as possible.

The position of the effector as seen (v) is updated in real time each time the camera sees the corresponding tag. The algorithm works by sending the inverse

[3] Here, as in [6], we use the terms *coordinate frame* and *pose* interchangeably.

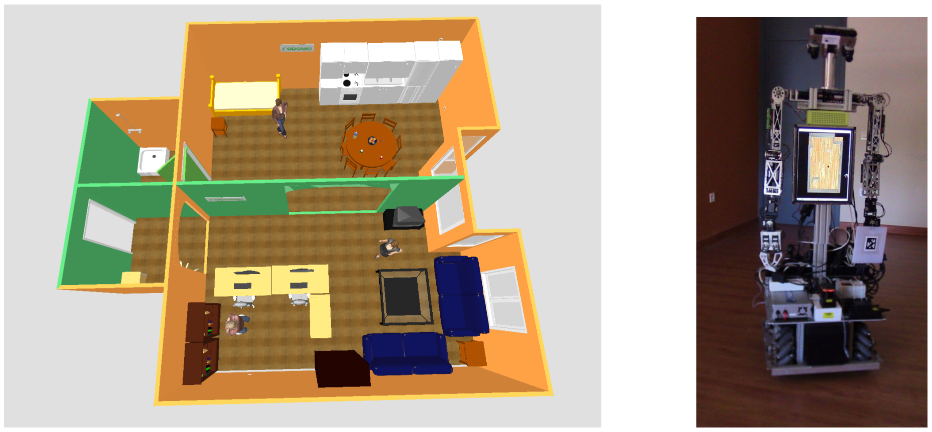

(a) RoboHome (b) Ursus 3

Fig. 6. (a) Autonomy Lab in RoboLab, UEx. A 70m apartment where robots can interact with people in HRI experiments. b)Ursus is the third generation of RoboLab's mobile manipulator [21]. It has a head, two 7 DOFs arms and an omnidirectional base custom built with Mecanum wheels.

kinematics module a series of intermediate positions (k_n) of the effector as seen by the camera (v) and as close to the target (t) as possible.

The algorithm works as follows:

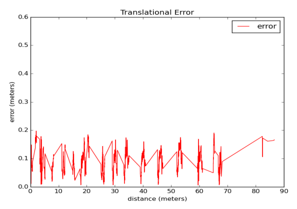
1. The IK target k is initialized as the actual target: $k = t$
2. The translation and rotational *errors* between the visual position and the IK position are calculated: ${}^{t}t_v$ and ${}^{t}R_v$, respectively.
3. The algorithm successfully stops if the errors are small, or unsuccessfully if it has been running for a long period of time.
4. A corrected IK position (k_i) so that: ${}^{k_i}t_t = -{}^{v_i}t_t$ and ${}^{k_i}R_t = {}^{v_i}R_t{}^{-}1$ are calculated and sent to the IK module.
5. If $error > threshold$ go to step 2.

3 Experiments

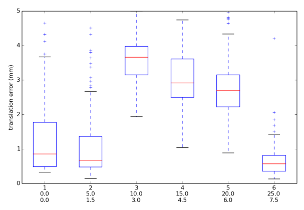
Two experiments have been made to test and validate the current state of the new navigation agent within the robot Ursus, shown in Figure 6. Also in the same Figure it is depicted a 3D drawing of RoboLab's Autonomy Lab, a $70m^2$ apartment conditioned for social robotics research 6.

3.1 Localization and navigation experiment

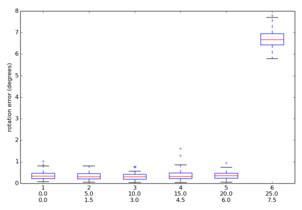
The first experiment was designed to measure the robustness of the CGR localization algorithm in real world, highly perturbing conditions. A list with the 2D segments representing the walls and furniture on the floor was obtained from the construction blueprint and by manual measurement. The robot was sent to a set of random locations in the apartment and the ground truth position was recorded using a set of AprilTags [22]. Figure 7 shows the evolution of the accumulated error during 90 meters of unstopped navigation. The mean is 8cm and the standard deviation 3cm showing a very good localization performance for this kind of social tasks.



(a) Accumulated error



(b) Translation error



(c) Rotation error

Fig. 7. (a) Accumulated error of the robot after 90m of continuous navigation in a noisy environment, with unmodeled objects and people walking around. Mean 0.08 m. Variance 0.0014. Standard deviation 0.0367. b)The y axis shows the errors (euclidean norm) obtained in translation. The X axis shows the experiment number and the standard deviation of the errors. c) The Y axis shows the errors obtained in rotation. The X axis shows the experiment number and the standard deviation of the errors.

3.2 Arm control experiment

A simulated experiment with increasing synthetic calibration errors was conducted to show how the visual servoing algorithm is capable of moving the effector to target poses with good precision. Six experiments were performed sending the arm to 100 different positions in each one. An increasing level of error was injected for each experiment in the arm's kinematic configuration. Errors were assigned in increments of 1.5 degrees for joint angular position (encoder error) and of 5 mm for translational joint position (mechanical misalignment). The 100 targets of each experiment were randomly generated from a work space of dimensions $X \in 140 - 130$ $Y \in 780 - 800$ and $Z \in 300 - 900$ in front of the robot, where $+Z$ points outwards perpendicular to robot's chest. The robot's arm movement would stop whenever the rotational error was lower than 0.1 rad and the translational error was lower than 5 mm. Errors were recorded at the end of each robot's arm movement. As can be seen in Figure 7, even in the presence of very high calibration errors the end effector ended with translation errors under the 5 mm threshold, and rotational errors were widely bellow the 0.1 rad. threshold except for the most adverse situation in which the mean settled close to 8 degrees.

4 Conclusions

In this paper we have shown work in progress on the construction of a new navigation software agent for the RoboCog architecture. Five interrelated functionalities have been reassessed and new algorithms have replaced the existing ones. As a result we have now a very robust localization component that can take pose estimations of the robot from a number of other components and maintain a reliable pose for extended periods of time under real life aggressive conditions. In this group, CGR provides map based localization and its performance is excellent. Robot navigation was formerly based on the combination of RRT and VFH+ and their integration presented important problems. With the introduction of a persistent structure coding the current path, those problems no longer exist and a whole new set of new possibilities bounce into. The path allows for smoother paths, easier recovery, better controllers and a simpler and more rational software architecture. As a future line of work, we want to enrich the path with semantic annotations, leading also to the idea of semantic path planning and execution. Arm planning has also received an important improvement with this work. The early problems that we had with very low repeatability and precision during grasping operations, have been reduced to a point where we can grab a cup with much more reliability. The decomposition of arm grabbing gestures in two actions, one driven only by internal feedback and the other by visual feedback in the camera space, is a promising line of future research, markless visual servoing being the next challenge.

Acknowledgements

This paper has been partially supported by the Spanish Ministerio de Economía y Competitividad TIN2012-TIN2012-38079-C01, ADAPTA project of the INTERCONNECTA program and FEDER funds

References

1. Sean Quinlan and Oussama Khatib. Elastic bands: connecting path planning and control. *[1993] Proceedings IEEE International Conference on Robotics and Automation*, 1993.
2. Rebeca Marfil, Luis V Calderita, Juan Pedro Bandera, Luis J Manso, and Antonio Bandera. Toward social cognition in robotics : Extracting and internalizing meaning from perception. In *Workshop of Physical Agents WAF2014*, number June, pages 1–12, 2014.
3. L. V. Calderita, P. Bustos, C. Suárez Mejías, F. Fernández, and a. Bandera. Therapist: Towards an autonomous socially interactive robot for motor and neurorehabilitation therapies for children. *7th International Conference on Pervasive Computing Technologies for Healthcare and Workshops*, 1:374–377, 2013.
4. O. Brock and O. Khatib. Elastic strips: A framework for motion generation in human environments. *The International Journal of Robotics Research*, 21(12):1031–1052, 2002.

5. Yuandong Yang and Oliver Brock. Elastic roadmaps - motion generation for autonomous mobile manipulation. *Autonomous Robots*, 28(1):113–130, 9 2009.
6. S. Hutchinson, G.D. Hager, and P.I. Corke. A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, 12(5):651–670, 1996.
7. B Espiau, F Chaumette, and Patrick Rives. A new approach to visual servoing in robotics. *Robotics and Automation, IEEE Transactions on*, 8(3):313–326, 1992.
8. Ezio Malisfranc, O I S Chaumette Irisa, Inria Rennes, and France Received. 2 1/2 d visual servoing with respect to unknown objects through a new estimation scheme of camera displacement. *International Journal of Computer Vision*, 37(1):79–97, 2000.
9. Trumpington St. Application of lie algebras to visual servoing. *International Journal of Computer Vision*, 37(1):21–41, 2000.
10. Eric T Baumgartner, Michael J Seelinger, Martin Fessler, Andreas Aldekamp, Emilio Gonzalez-galvan, John-david Yoder, Steven B Skaar, and Notre Dame. Accurate 3-d robotic point positioning using camera space manipulation. pages 2–5.
11. Xavi Gratal, Javier Romero, Jeannette Bohg, and Danica Kragic. Visual servoing on unknown objects. *Mechatronics*, 22(4):423–435, 2012.
12. Michael Sapienza, Miles Hansard, and Radu Horaud. Real-time visuomotor update of an active binocular head. *Autonomous Robots*, (July 2011), 9 2012.
13. Luis J Manso, Luis V Calderita, Pablo Bustos, Javier Garc, and Fernando Fern. A general-purpose architecture to control mobile robots. *Waf2014*, (June):1–12, 2014.
14. Luis Manso, Pilar Bachiller, Pablo Bustos, and Luis Calderita. Robocomp: a tool-based robotics framework. In *SIMPAR, Second International COnference on Simulation, Modelling and Programming for Autonomous Robots*, 2010.
15. J. Biswas, B. Coltin, and M. Veloso. Corrective gradient refinement for mobile robot localization. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 73–78, 9 2011.
16. Edwin Olson. Apriltag: A robust and flexible visual fiducial system. *Proceedings - IEEE International Conference on Robotics and Automation*, pages 3400–3407, 2011.
17. Steven M LaValle. Planning algorithms. *Methods*, 2006(2):842, 2006.
18. L. E. Kavraki, P. Svestka, JC Latombe, and M. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.
19. Ioan a. Şucan, Mark Moll, and Lydia Kavraki. The open motion planning library. *IEEE Robotics and Automation Magazine*, 19(4):72–82, 2012.
20. S. Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93109, 2000.
21. F Martín, J. Mateos, F. J. Lera, P. Bustos, and V Matellán. A robotic platform for domestic applications. In *WAF 2014*, number June, pages 1–8, 2014.
22. E. Olson, J. Leonard, and S. Teller. Fast iterative alignment of pose graphs with poor initial estimates. *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, (May):2262–2269, 2006.
23. Kavraki, L. E. Svestka, P. Latombe, JC Overmars, M., Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces, IEEE Trans. Robot. Autom., vol. 12, no. 4, pp. 566580, 1996.
24. Chaumette F., Hutchinson S., Visual Servo Control (II), IEEE Robotics and Automation Magazine, March, 2007.