Integrating Path Planning and Visual Servoing in Manipulation Tasks

Joseph Mirabel, Alexis Nicolin, Florent Lamiraux, Olivier Stasse, Sébastien Boria

Integrating Path Planning and Visual Servoing in Manipulation Tasks

Joseph Mirabel[1], Alexis Nicolin[1,2], Florent Lamiraux[1] , Olivier Stasse[1] and Sébastien Boria[2]

***Abstract*— This paper proposes a novel methodology to integrate both geometric manipulation planning and visual servoing. Geometric manipulation planning is able to handle robots with many degrees of freedom in cluttered environments while visual servoing makes execution of the planned trajectories robust to inaccuracies. The method is illustrated by a demonstration on a real humanoid robot manipulating an object.**

I. Introduction

Performing manipulation tasks including regrasping on a humanoid robot is known to be a difficult task for several reasons. The first reason is the necessity for the robot to measure the position of the objects to manipulate with onboard sensors, before grasping but also after each grasp or regrasp since the object may slightly slide in the gripper during those steps. This perception issue is far from being solved for any object. The second reason resides in the high number of degrees of freedom of the system (robot + objects) and in the complexity of manipulation planning in the presence of obstacles.

To cope with the first reason mentioned above, a lot of work has been proposed in visual servoing in order to control the motion of the robot based on the perception of the position of the objects that are to be manipulated.

To cope with the second reason, a lot of work has also been proposed to plan manipulation motions based on models of the robot, of the objects and of the obstacles.

To our knowledge however these two corpuses of work have largely evolved independently. Visual servoing usually aims at reducing an error between perceived and desired features. It does not take into account the reference trajectory that may have been planned priorly by a motion planner. One may argue that the planned trajectory implicitely defines a time-parameterized trajectory for the desired feature to be controlled by visual servoing, as in [15]. However, if the feature image is affected only by a subset of the robot degrees of freedom – the left arm for instance, the other degrees of freedom will not be controlled.

A. Visual servoing

[3], [4] provide an overview of the state of the art in this domain until 2007. The second reference mentions the case of moving target only at the very end. In a more recent work [2], the authors perform a grasp task on a Romeo humanoid robot, tracking the position of the hand and of the object to grasp with a camera. In this latter work, there are no obstacles and collision of the hand with the object during the approach phase is not addressed.

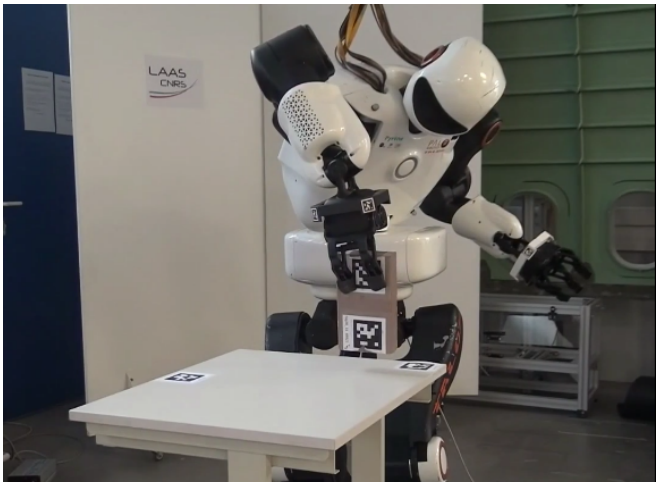

Fig. 1. Example of the effect of visual servoing on the motion execution accuracy. Eventhough the box has slipped in the gripper, the controller successfully positions the box vertically before putting it on the table.

B. Manipulation planning

Manipulation planning is a class of path planning problems where some robots manipulate some objects with grippers, in an environment cluttered with obstacles. Manipulation constraints raised by the fact that objects cannot move if they are not grasped and that they are ridigly attached to a gripper when they are grasped define several foliations of the configuration space of the whole system (robots + objects). From a given configuration with a given set of grasps, the system can only move on a submanifold of the configuration space called a leaf.

Since the pioneering paper [24], manipulation planning has been tackled using random sampling based method [10], [12] in order to explore the leaves of those foliations [21]. Manipulation planning is traditionally decomposed into several subdomains: Rearrangement planning [11], [20], [13], [6] where the goal specifies only the final position of the objects, multi-arm motion planning [8], [9], [5], and navigation among movable obstacles.

C. Sensor-based manipulation planning and control

[25] addresses the problem of rearrangment planning for a disc robot manipulating circular objects in the plane. The paper presents impressive experimental results with a quadruped robot equipped with a lidar and avoiding

This work has been supported by the Airbus-CNRS Joint laboratory Rob4Fam.

[1]LAAS, University of Toulouse, CNRS, Toulouse, France
[2]Airbus SAS,OMIR, Toulouse, France

unexpected obstacles. [22] proposes a method to plan a sequence of controllers that move a system composed of two robots and one object from an initial state to a goal state in a dynamic environment. The algorithm performs a random exploration of the configuration space, using simulated controllers as a steering method. The controllers are implemented using eTaSL/eTC framework [1]. Our work share similarities with this latter paper. The main differences are that we do not explicitely address dynamic environments, but we implement visual servoing.

D. Contribution

In this paper, we propose a novel approach to manipulation planning and control for a robot with a high number of degrees of freedom in the presence of obstacles. The approach is broken down into three steps. In the first step, a manipulation planning algorithm computes a manipulation planning path starting from the initial configuration and reaching a goal configuration or set of configurations. In the second step, the manipulation path is decomposed into segments. To each segment a controller taking into account both the planned path and visual features attached to objects or robot bodies is associated. These controllers regulate sequences of tasks with priority orders [23]. In the third step, a finite state machine keeps track of the current segment and selects the controller accordingly. Note that a preliminary version of this work has been described in [19]. However, in this previous work, no visual servoing was implemented.

The paper is organized as follows. In Section II, we define the problem and introduce some notation. In Section III, we describe the manipulation planning algorithm that we use to produce reference trajectories. In Section IV, we explain how a manipulation trajectory is mapped to a sequence of hierarchical task based controllers. In Section V, we describe how the resulting motion is executed by the robot. In Section VI, we describe some experimental results that validate the approach.

II. Definition of the problem

We consider a robot with configuration space denoted as $\mathcal{C}_r$ and $n_o \geq 1$ objects of configuration spaces $SE(3)$[1]. The configuration space of the whole system is the Cartesian product of the configuration spaces of the robot and objects: $\mathcal{C} = \mathcal{C}_r \times SE(3)^{n_o}$. Some static obstacles are present in the workspace of the robot.

A. Grippers, handles, grasps

The robot is equipped with $n_g \geq 1$ *grippers i.e.* frames that are attached to some of its links. To each object o_i, $i \in \{1, \cdots n_o\}$, $n_{h,i} \geq 0$ frames called *handles* are rigidly attached. We denote by

- $g_1, \cdots, g_{n_g}$ the robot grippers,
- $h_{j,i}$ for $i \in \{1, \cdots, n_o\}$ and $j \in \{1, \cdots, n_{h,i}\}$ the j-th handle of o_i.

[1] $SE(3)$ is the group of rigid-body transformations

For any configuration $\mathbf{q} \in \mathcal{C}$, and any gripper g and handle h, we denote by $g(\mathbf{q}) \in SE(3)$ and $h(\mathbf{q}) \in SE(3)$ the positions of g and h in configuration $\mathbf{q}$. We define a *grasp* – denoted as $gr(g,h)$ – as the part of $\mathcal{C}$ containing all configurations of the system such that the frames defined by g and h are at the same position:

$$gr(g,h) = \{\mathbf{q} \in \mathcal{C},\ g(\mathbf{q}) = h(\mathbf{q})\}.$$

B. Contact surfaces, placements

To each object o_i, $i \in \{1, \cdots n_o\}$, $n_{c,i} \geq 0$ convex polygons called *object contact surfaces* are rigidly attached. Convex polygons called *environment contact surfaces* are attached to the environment.

- Contact surfaces of o_i are denoted by $c_{i,j}$ for $j \in \{1, \cdots, n_{c,i}\}$.
- Environment contact surfaces are denoted by $c_{0,j}$, for $j \in \{1, \cdots, n_{c,0}\}$.

An object is in a *placement* configuration if one of its contact surfaces is in contact with one environment contact surface. Two convex polygons are said to be in contact if the plane supporting them is the same and if the center of the first polygon is inside the second polygon. Contact constraints can be represented by an implicit numerical constraint over the configuration $\mathcal{C}$. See [17] Section 4.1.2 for details.

We denote as $pl(o_i)$ the subset of $\mathcal{C}$ of configurations where object o_i is in a placement configuration.

C. Visual features

The robot is equipped with a RGB-D camera providing an image $I \in RGBD = Co \times Co \times Co \times \mathbb{R}$ where $Co \triangleq \{0, \cdots, 255\}$. We assume that a localization function is able to localize object o_i with $i \in \{1, \cdots, n_o\}$ or gripper g_j with $j \in \{1, \cdots, n_g\}$ such that:

$$\begin{array}{l} {}^c\hat{T}_{o_i} : I_{o_i} \mapsto SE(3) \\ {}^c\hat{T}_{g_j} : I_{g_j} \mapsto SE(3) \end{array} \tag{1}$$

where $I_{o_i} \subset RGBD$ and $I_{g_j} \subset RGBD$ are the domains of the image space where the localization function successfully computes a pose, ${}^c\hat{T}_{g_j}$, ${}^c\hat{T}_{o_i}$ are the respective poses of g_j and o_i in the camera frame, as computed by the localization function. We denote by

$${}^{g_j}\hat{T}_{o_i} = {}^c\hat{T}_{g_j}^{-1}\ {}^c\hat{T}_{o_i} : I_{o_i} \cap I_{g_j} \mapsto SE(3) \tag{2}$$

the localization function that maps an image to the relative pose of the object in the gripper frame.

D. Problem definition

The problem we want to solve consists in moving the system from an initial configuration where the objects are in stable configurations, to a goal configuration where the objects are still in stable configurations. We decompose this problem in a sequence of sub-problems:

1) *manipulation planning:* computation of a collision-free manipulation path along which each object is either in a stable configuration or grasped by a gripper,

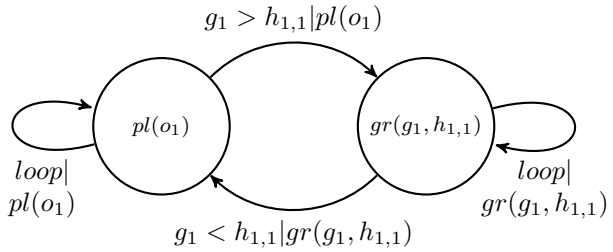

Fig. 2. Constraint graph corresponding to a robot with one gripper and an object with one handle. "$g > h$", (resp. "$g < h$") in transition names means that grasp of h by g is created (resp. removed). "$|s$" means that transition starts from state s. Note that transition "$g_1 > h_{1,1}|pl(o_1)$" lies in state $pl(o_1)$ and therefore contains the constraint that the object should be in a stable position. It also contains an additional constraint stating that the object should not move along this transition.

2) *controller generation:* computation of a sequence of controllers corresponding to successive segments of the planned path,
3) *motion execution:* successive activation of each controller after detection of the end of the previous segment.

Item 2) above is the main contribution of this paper. In the three following sections, we explain how we tackle the above sub-problems.

III. Manipulation planning

To plan a manipulation path, we use algorithm *Manipulation RRT* as described in [16]. This algorithm is an extension of RRT that explores the foliations induced by the manipulation constraints.

A. Constraint graph

The manipulation constraints are stored in a graph

- the nodes of which are sets of grasps as defined in Section II-A called *states*, and
- the edges of which are called *transitions* and connect states that differ by only one grasp.

Transitions contain the constraints of the state they belong to and additional constraints stating that objects not grasped should not move. See Figure 2 for an example.

B. Grasp waypoints

Grasp configurations are usually very close to collision since the object and gripper are almost in contact. To avoid the so called *narrow passage* effect in path planning, we define some intermediate states called waypoints in the constraint graph. *clearance* parameters are added in the model of each object handle and of each robot gripper. For each pair (g, h) of gripper and handle, the pregrasp of h by g is defined as the following subset of $\mathcal{C}$

$$\left\{\mathbf{q} \in \mathcal{C},\ g(\mathbf{q})^{-1} h(\mathbf{q}) = \mathbf{T}\right\},$$

where

- $g(\mathbf{q})^{-1} h(\mathbf{q}) \in SE(3)$ is the relative position of h in g frame (in configuration $\mathbf{q}$),

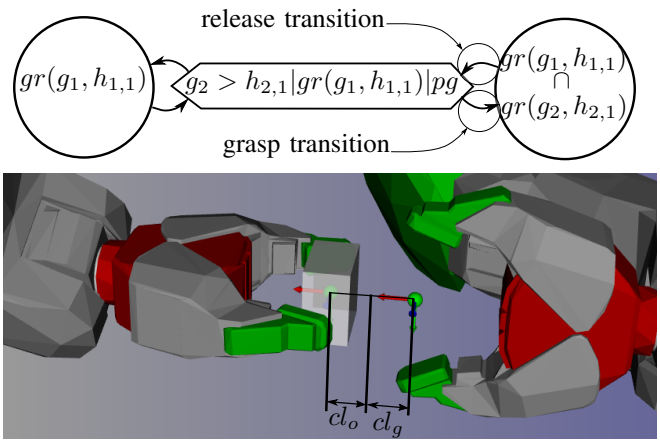

Fig. 3. Top: partial view of a constraint graph with a grasp waypoint state inserted between two states. The robot has two grippers g_1, g_2 and manipulates object 1 by two handles $h_{1,1}$, $h_{1,2}$. Bottom: example of a configuration in waypoint state $g_2 > h_{2,1}|gr(g_1, h_{1,1})|pg$. cl_o and cl_1 respectively represent the clearance of $h_{1,2}$ and of g_2. Red arrows represent x-axes.

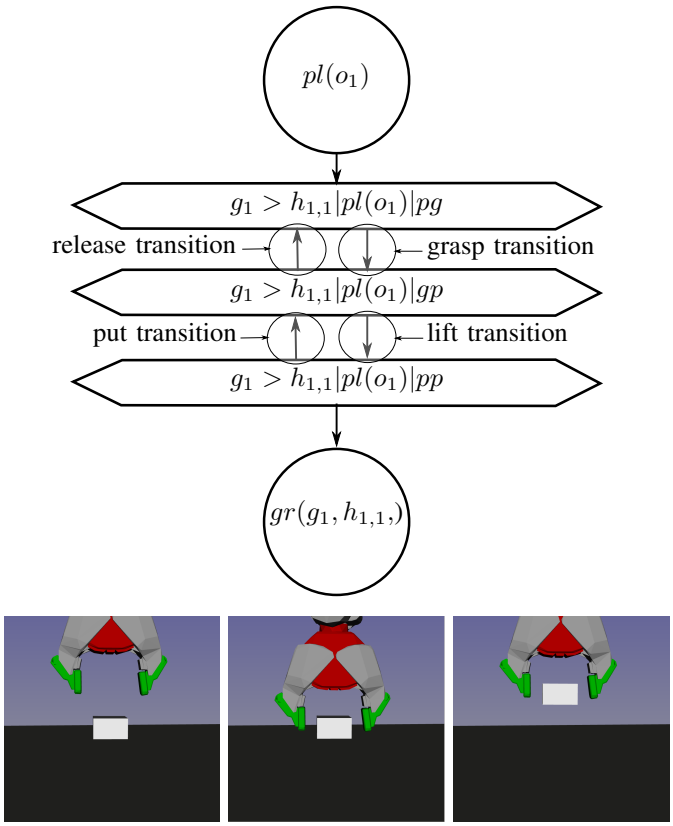

Fig. 4. Top: partial view of a constraint graph with 3 placement waypoint states inserted between two states. Bottom: example of 3 configurations in waypoint states *pregrasp* (pg), *grasp-placement* (gp) and *preplacement* (pp).

- $\mathbf{T} \in SE(3)$ is the translation along x axis of abscissa the sum of gripper and handle clearances.

With this definition, we augment the constraint graph with intermediate *pregrasp states* along transitions that create a new grasp. See figure 3 for an example.

C. Placement waypoints

Between a state where an object is in placement and a state where the same object is grasped – all other grasps and placement being unchanged, we insert 3 waypoint states corresponding to

- *pregrasp*: defined as in the previous section (III-B),
- *grasp-placement*: the object is grasped and still in placement,
- *preplacement*: the object is grasped and translated by a positive distance equal to the grasp clearance along the normal to the environment contact surface.

D. Special transitions

Using definitions in sections III-B and III-C, for clarity, we refer to some transitions as follows.

- *Grasp transitions* connect *pregrasp* and *grasp-placement* states.
- *Release transitions* connect *grasp-placement* and *pregrasp* states.
- *Lift transitions* connect *grasp-placement* and *preplacement* states.
- *Put transitions* connect *preplacement* and *grasp-placement* states.
- *Loop transitions* connect a state to itself.

In the above definitions, we do not make any difference between waypoint and regular states. Figures 3 and 4 show examples of these transitions.

E. State pruning

Without additional information, each gripper may grasp each handle, making the number of states potentially very high. To keep the number of states reasonable, we filter out states that are empty because

1) configurations satisfying the grasp constraints of the state are always in collision, or
2) no configuration satisfies the grasp constraints of the state.

States of the first type are detected by randomly generating one configuration in the state and by testing collision only of the bodies constituting the grippers. Those body relative positions are indeed constant for all configurations of the state. States are declared of the second type if none of a given number of attempts to generate a configuration in those states is successful. Attempting to generate a node in a state consists in generating a random configuration and to solve the state constraints using a Newton based non-linear equation solver as described in [18] (Algo 1).

F. Path smoothing

The result of *Manipulation RRT* is post-processed in two successive steps in order to make it executable on a real robot.

1) a *random shortcut* algorithm is applied [7] to shorten the path and remove useless detours,
2) the result of the previous step is time-parameterized using sequences of Bezier curves in order to make it continuously differentiable and in order to bound the maximal joint accelerations from above.

IV. Sensor based motion control

The output of the manipulation planning process described in the previous section is a continuously differentiable trajectory composed of a sequence of segments such that each segment is subject to a set of numerical constraints stored in the constraint graph edges. Segments are linked by configurations that belong to states (including waypoint states) of the constraint graph.

If models of the robot, of the object and of the environment were perfect, if the control of the robot motors were perfect and if the sensors were perfect, controlling the joints by feeding them with the planned trajectory would make the task execution successful. However, in the real world, none of the above is perfect. Particularly, if the robot links are light as is the case for legged humanoid robots, the overall flexibility of the structure induces important variations between the position of the end effectors computed via forward kinematics and the actual position. Those variations can only be partially corrected by a calibration step. In this paper, we use visual servoing to cope with the above inaccuracies.

To do so, we use a hierarchical task-based controller as described in [14]. This software basically implements the algorithm described in [23].

A. Hierarchical task based controller

The controller is based on the following concepts:

- *task:* a mapping from $\mathcal{C}$ to $\mathbb{T}$ a vector space or a Lie-group (usually $SE(3)$). It can represent the pose of an end effector, the posture of the robot, the pose of an object;
- *task reference:* a mapping from $\mathbb{R}$ to $\mathbb{T}$ that represents the desired trajectory of the task along time.
- *error:* a mapping from $\mathbb{T}\times\mathbb{T}$ to a vector space that maps to zero pairs of identical task values. For instance, if the task is the pose ($\mathbb{T} = SE(3)$) of a robot end effector, and if T_1, T_2 are two elements of $SE(3)$,

$$\mathbf{e}_{SE(3)}(T_1, T_2) = \log\left(T_1^{-1} T_2\right) \tag{3}$$

is the screw velocity $(v, \omega) \in \mathbb{R}^6$ that moves T_1 to T_2 in unit time. Particularly, the error is equal to zero iff $T_1 = T_2$. If $\mathbb{T} = \mathbb{R}^n$ for a given positive integer n,

$$\mathbf{e}_{\mathbb{R}^n}(T_1, T_2) = T_2 - T_1. \tag{4}$$

Note that expression (3) is equivalent to (4) if we consider $\mathbb{R}^n$ as a Lie group with $+$ operator.

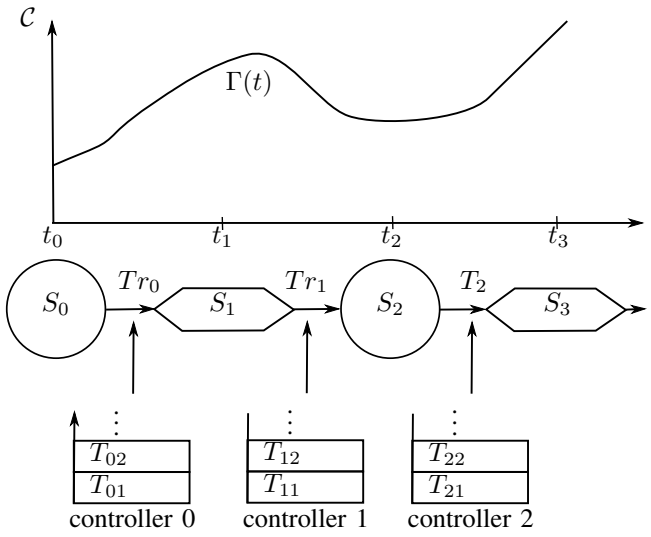

Fig. 5. Sequence of controllers. As explained in Section III, the output of the manipulation planning algorithm is a sequence of time-parameterized trajectory segments belonging to transitions of the constraint graph. In this figure, all states – waypoint or not – are represented by circles. A hierarchical task based controller is associated to each transition.

If we consider a planned trajectory Γ as a mapping from $\mathbb{R}$ to $\mathcal{C}$, and a task T with output space $\mathbb{T}$, we define the *task reference associated to* T for the trajectory Γ as the mapping from $\mathbb{R}$ to $\mathbb{T}$:

$$T^*(t) \triangleq T(\Gamma(t)) \tag{5}$$

and the *error for this task along* Γ as the mapping from $\mathcal{C}\times\mathbb{R}$ to $\mathbb{R}^n$ for a given n:

$$\mathbf{e}(\mathbf{q}, t) \triangleq \mathbf{e}_{\mathbb{T}}(T(\mathbf{q}), T^*(t)). \tag{6}$$

A hierarchical task based controller is initialized with an ordered list of tasks T_i, $i \in \{1, \cdots, m\}$ for some positive integrer m, with decreasing order of priority. It takes as input a robot configuration $\mathbf{q}$ and computes a robot control $\dot{\mathbf{q}}$ in order to make the errors converge to 0. If no value of $\dot{\mathbf{q}}$ makes all the errors converge to 0, the controller computes the best control value $\dot{\mathbf{q}}$ given the priority order. We refer to [14] for details about the computations.

B. Synthesis of a sequence of controllers

The main contribution of this paper is the automatic generation of a sequence of hierarchical task-based controllers, one for each transition of the constraint graph along a planned trajectory. To each type of transitions defined in Section III-D, we associate a controller as described below.

a) System specific task: Possible robot specific constraints are inserted at the higher priority level (level 1). This task can be for instance quasi-static equilibrium of a legged humanoid robot, or a predefined trajectory of the base of a mobile manipulator. Note that these constraints are also used during motion planning.

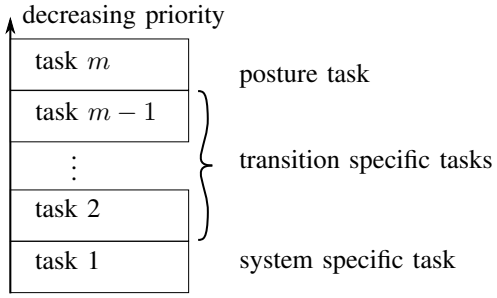

Fig. 6. Controllers are associated to each type of transition.

b) Posture task: The posture task takes values in $\mathbb{R}^n$ where n is the number of actuated degrees of freedom of the robot. The value of the task for a given configuration $\mathbf{q}$ is obtained by extracting from $\mathbf{q}$ the parameters corresponding to the actuated degrees of freedom.

Tasks specific to each transition are inserted between the system specific task and the posture task as described in Figure 6.

c) Visual servoing task: In this task, the relative pose of object i in the robot gripper g_j frame is regulated. Therefore the task output is $T_{g_j}^{-1} T_{o_i}$ where T_{g_j} and T_{o_i} are the respective poses of g_j and of o_i. Following (3), the error of the task is defined as:

$$\mathbf{e} = \log\left(({}^{o_i}\hat{T}_{g_j})^{-1} T_{g_j}^{*-1} T_{o_i}^* \right) \tag{7}$$

where ${}^{o_i}\hat{T}_{g_j}$, defined by (2) is the measured value of the task, while $T_{g_j}^{*-1} T_{o_i}^*$ is the time varying reference of the task.

d) Grasp and release transitions: if the link of the gripper about to grasp and the object to be grasped are equipped with a visual feature, the transition specific task is a visual servoing task of output space $SE(3)$. The value of the task is the relative pose of the object in the gripper frame.

e) Put and Lift transitions: if the object about to be put and the object (or environment) holding the contact surface are equipped with a visual feature, the transition specific task is a visual servoing task of output space $SE(3)$. The value of the task is the relative pose of the object in the frame of the contact surface.

f) Loop transitions: if along the transition, an object is grasped by two grippers, the transition specific task is the relative pose of a gripper with respect to the other one. This task value can be measured

- by visual features if available, or
- joint encoders and forward kinematics. In this case, the reference of the task is not the value computed from Equation (5), but the value measured by the joint encoders at the beginning of the transition.

g) Other transitions: transitions that do not fit one of the above definitions do not have any specific tasks.

h) Note: visual servoing tasks are inserted only if the corresponding visual features are visible from the robot

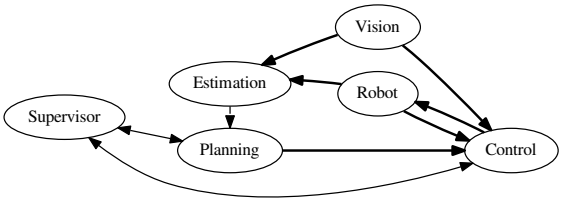

Fig. 7. ROS Architecture: arrows display the flow of information.

camera. This can be enforced at the motion planning level by adding a constraint on the camera orientation, or at least predicted from the planned trajectory.

V. Motion execution

Motion execution is performed on the robot using ROS architecture. The ROS nodes involved are displayed in Figure 7 and described below.

a) Vision: detects and publishes the poses of visual features with respect to the camera frame.

b) Estimation: reads the poses of objects as published by the *Vision* and the robot joint encoders. These raw data correspond to a configuration $\hat{\mathbf{q}}$ that is not contained in any state of the constraint graph. The node projects $\hat{\mathbf{q}}$ on all states and publishes the state and configuration $\hat{\mathbf{q}}_{proj}$ corresponding to the closest projection of $\hat{\mathbf{q}}$.

c) Planning: runs the manipulation planning algorithm and publishes the references that are extracted from the solution trajectory and that the controllers request. The initial configuration is read from the *Estimation* node.

d) Supervisor: computes the sequence of controllers and handles synchronisation between the different nodes.

e) Control: embeds the hierarchical task based controllers into the **ros_control** interface.

VI. Experimental results

We have implemented our framework on Pyrene robot displayed in Figure 1. The experimental setup is composed of the humanoid robot, a table and a wooden box both equipped with AprilTags [26].

In this section, we show the execution of a manipulation sequence where the robot is asked to flip a box upside down on a table. The reference path is planned and optimized by our software platform `HPP` as explained in section III.

VII. Conclusion and perspectives

This article proposes a new framework to automatize the bridge between motion planning and control. This is achieved by extracting in the planned trajectory the references that are relevant for a given portion of the trajectory. This article illustrates the application of this principle to visual servoing. Experimentations on Pyrene robot validate the effectiveness of the approach. They also point out some possible improvements for a future work. The first improvement would consist in growing obstacles and robot bodies for

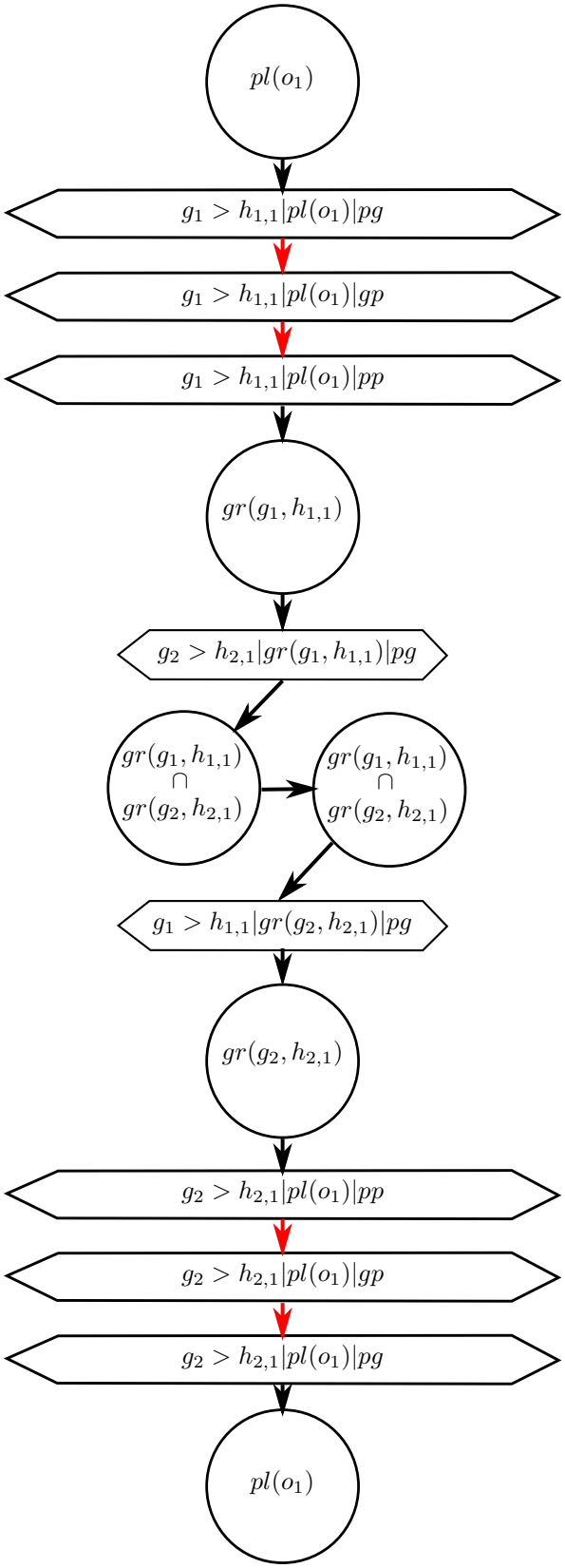

Fig. 8. Sequence of states visited to flip the box upside down. The robot grasps with one hand, changes hand and releases the object in a stable position. Red arrows correspond to visual servoing transitions. Note that visual servoing was not performed when changing hand since the required tags were not all in the field of view of the camera.

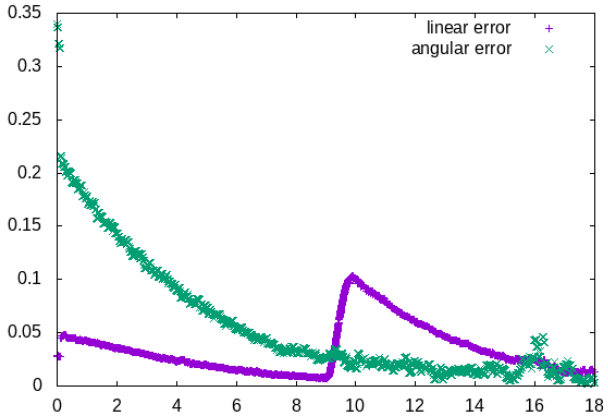

Fig. 9. Plots of the linear and angular errors of the visual servoing task along the approaching box motion (first red transition in Figure 8). The angular error decreases exponentially. The linear error decreases, then increases between 9 and 10s and then decreases exponentially again. The increase is probably due to the descending motion of the gripper to grasp the box. The gain of the visual servoing task is low because of the small frequency of the visual servoing process. This implies a delay in following the descending trajectory.

manipulation planning to account for poor robot calibration. This operation would be performed only along transitions without visual servoing. The second one would consist in providing feedforward to the visual servoing tasks in order to better follow moving targets.

REFERENCES

[1] E. Aertbelin and J. De Schutter, "Etasl/etc: A constraint-based task specification language and robot controller using expression graphs," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 09 2014.

[2] D. J. Agravante, G. Claudio, F. Spindler, and F. Chaumette, "Visual servoing in an optimization framework for the whole-body control of humanoid robots," *IEEE Robotics and Automation Letters*, vol. 2, no. 2, pp. 608–615, Apr. 2017. [Online]. Available: https://hal.inria.fr/hal-01421734

[3] F. Chaumette and S. Hutchinson, "Visual servo control, part i: Basic approaches," *IEEE Robotics and Automation Magazine*, vol. 13, no. 4, pp. 82–90, December 2006.

[4] ——, "Visual servo control, part ii: Advanced approaches," *IEEE Robotics and Automation Magazine*, vol. 14, no. 1, pp. 109–118, March 2007.

[5] A. Dobson and K. Bekris, "Planning representations and algorithms for prehensile multi-arm manipulation," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Hamburg, Germany, 2015.

[6] C. R. Garrett, T. Lozano-Pérez, and L. P. Kaelbling, "Ffrob: Leveraging symbolic planning for efficient task and motion planning," *CoRR*, vol. abs/1608.01335, 2016. [Online]. Available: http://arxiv.org/abs/1608.01335

[7] R. Geraerts and M. Overmars, "Creating high-quality paths for motion planning." *International Journal of Robotics Research*, vol. 26, no. 8, pp. 845–863, 2007.

[8] M. Gharbi, J. Cortés, , and T. Siméon, "Roadmap composition for multi-arm systems path planning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Saint-Louis, USA, 2009.

[9] K. Harada, T. Tsuji, and J.-P. Laumond, "A manipulation motion planner for dual-arm industrial manipulators. in proceedings of," in *IEEE International Conference on Robotics and Automation*, Hongkong, China, 2014, pp. 928–934.

[10] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, August 1996.

[11] A. Krontiris and K. Bekris, "Dealing with difficult instances of object rearrangement," in *Robotics Science and Systems*, Roma, Italy, 2015.

[12] S. M. Lavalle and J. J. Kuffner, "Randomized kinodynamic planning," *International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, May 2001.

[13] P. Lertkultanon and Q.-C. Pham, "A single-query manipulation planner," *IEEE Robotics and Automation Letters*, vol. 1, no. 1, pp. 198–205, 2015.

[14] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks," in *International Conference on Advanced Robotics (ICAR)*, 2009.

[15] Y. Mezouar and F. Chaumette, "Path planning for robust image-based control," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 4, pp. 534–549, 2002. [Online]. Available: https://hal.inria.fr/inria-00352101

[16] J. Mirabel and F. Lamiraux, "Manipulation planning: Addressing the crossed foliation issue," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2017. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01358767

[17] J. Mirabel, "Manipulation planning for documented objects," Theses, Institut National Polytechnique De Toulouse, Feb. 2017. [Online]. Available: https://hal.laas.fr/tel-01516897

[18] J. Mirabel and F. Lamiraux, "Handling implicit and explicit constraints in manipulation planning," in *Robotics: Science and Systems*, Pittsburg, USA, June 2018. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01804774

[19] A. Nicolin, J. Mirabel, S. Boria, O. Stasse, and F. Lamiraux, "Agimus: a new framework for mapping manipulation motion plans to sequences of hierarchical task-based controllers," in *IEEE/SICE International Symposium on System Integration*, Honolulu, United States, Jan. 2020. [Online]. Available: https://hal.laas.fr/hal-02466543

[20] J. Ota, "Rearrangement of multiple movable objects-integration of global and local planning methodology," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, vol. 2, 2004, pp. 1962–1967.

[21] P. S. Schmitt, W. Neubauer, W. Feiten, K. M. Wurm, G. V. Wichert, and W. Burgard, "Optimal, sampling-based manipulation planning," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 3426–3432. [Online]. Available: http://ais.informatik.uni-freiburg.de/publications/papers/schmitt17icra.pdf

[22] P. S. Schmitt, F. Wirnshofer, K. M. Wurm, G. V. Wichert, and W. Burgard, "Modeling and planning manipulation in dynamic environments," in *2019 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2019. [Online]. Available: http://ais.informatik.uni-freiburg.de/publications/papers/schmitt19icra.pdf

[23] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *International Conferenceon Advanced Robotics*, 1991, pp. 1211–1216.

[24] T. Siméon, J.-P. Laumond, J. Cortés, and A. Sahbani, "Manipulation planning with probabilistic roadmaps," *International Journal of Robotics Research*, vol. 23, no. 7/8, July 2004.

[25] V. Vasilopoulos, T. Topping, W. Vega-Brown, N. Roy, and D. Koditschek, "Sensor-based reactive execution of symbolic rearrangement plans by a legged mobile manipulator," in *2018 IEEE/RSJ International Conference on Intelligent Robotic Systems (IROS)*, 10 2018.

[26] J. Wang and E. Olson, "AprilTag 2: Efficient and robust fiducial detection," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2016.