Human Robot Interaction

Daniel Sidobre, Xavier Broquère, Jim Mainprice, Ernesto Burattini, Alberto Finzi, Silvia Rossi, Mariacarla Staffa

Human Robot Interaction

Daniel Sidobre[1], Xavier Broquère[1], Jim Mainprice[1],
Ernesto Burattini[2], Alberto Finzi[2], Silvia Rossi[2],
Mariacarla Staffa[3]

Abstract

To interact with humans, robots will possess a software architecture much more complete than current robots and be equipped with new functionalities. The purpose of this chapter is to introduce some necessary elements to build companion robots that interact physically with humans and particularly for the exchange of object tasks.

To obtain soft motion acceptable by humans, we use trajectories represented by cubic functions of time that allow mastering and limiting velocity, acceleration and jerk of the robot in the vicinity of the humans. During a hand-over task and to adapt its trajectory to the human behavior, the robot must adjust the time motion law and the path of the trajectory in real time. The necessity of real time planning is illustrated by the task of exchanging an object and in particular by the planning of double grasps. The robot has to choose dynamically a consistent grasp that enables both robot and human to hold simultaneously the exchanged object.

Then, we present a robotic control system endowed with attentional models and mechanisms suitable for balancing the trade-off between safe human-robot interaction (HRI) and effective task execution. In particular, these mechanisms allow the robot to increase or decrease the degree of attention toward relevant activities modulating the frequency of the monitoring rate and the speed associated to the robot movements. In this attentional framework, we consider pick-and-place and give-and-receive attentional behaviors. To assess the system performances we introduce suitable evaluation criteria taking into account safety, reliability, efficiency, and effectiveness.

[*1] Daniel Sidobre, Xavier Broquère, Jim Mainprice
LAAS-CNRS, 7, Av. Col. Roche, 31077 Toulouse France
daniel.sidobre@laas.fr, xavier@broquere.fr, jim.mainprice@laas.fr

[†2] Ernesto Burattini, Alberto Finzi, Silvia Rossi
DSF, Univ. di Napoli "Federico II", Via Cintia - 80126
{ernb,finzi,srossi}@na.infn.it

[‡3] Mariacarla Staffa
DIS, Univ. di Napoli "Federico II", Via Claudio, 21 - 80125
mariacarla.staffa@unina.it

1 Introduction

Until very recently, it was impossible to consider humans and robots living together. But now, robots start to become companions or co-workers of humans, opening an important research domain to build safe and intuitive cooperation. In this chapter we intend to introduce some elements to build such robots that are able to intuitively interact with humans.

In the context of Human Robot Interaction (HRI), intuitive and natural exchanges of objects between robots and humans represent a canonical task. But as we will see, such a robot system is much more complex than the current industrial robots that repeat the same tasks separated from humans by cages. We present some bricks that are necessary to give to the robots the necessary autonomy to react to the human motions and behavior. We focus the presentation on three key points. Firstly the necessity for a software architecture to coordinate and synchronize the different pieces of software. Then, we details the importance of the geometric reasoning in the case of the dynamic choice of a double grasp. In fact, to exchange an object, both the robot and the human must grasp simultaneously the object. So the robot must adapt its grasp and its motion to the human behavior in real time.

This introduces the importance of motion, which is then addressed from the geometric aspect of paths to the kinematic aspect of trajectories. Usually, motion planners compute a path, which is then executed by the robot controller, generally at a constant speed or across a dynamic simulator. But in both cases the time evolution is not taken into account at the planning level. For real time interaction with humans, the robot must master its time evolution and control where and when it hands over an object. To do this, we propose to integrate a simple model of trajectories based on series of cubic functions in a more standard random motion planner.

Finally, the human motions and the external environment should be continuously monitored by the robotic system looking for interaction opportunities while avoiding dangerous and unsafe situations. In this context, attentional mechanisms can play a crucial role: they can direct sensors towards the most salient sources of information, filter the available sensory input, and provide implicit sensory-motor coordination mechanisms to orchestrate and prioritize concurrent activities. In this work, we propose to deploy an attentional system to modulate the robotic arm motion and perception. The attentional system is expected to monitor and regulate multiple concurrent activities in order to achieve an effective coordination and interaction with the human movements in the operative space. We assume a frequency-based model of the executive attention, where each behavior is endowed with an adaptive internal clock that regulates the sensing rate and action activations. The frequency of sensor readings is here interpreted as a degree of attention towards a behavior: the higher the clock frequency, the higher the resolution at which the behavior is monitored and controlled. In this context, we consider attentional models for pick and place, give and receive, search and track (humans and salient objects).

2 Software Architecture

Clearly, to interact with humans, robots must be able to adapt in real time their movements to the behavior of the humans. Moreover, the robots must ensure safety and comfort for the humans all the while realizing socially acceptable movements. As tasks are not entirely defined in advance, but computed and adapted in real time, the robot must have all the software components to compute and adapt all the elements of an interactive manipulation task from supervision and task planning level to the hardware control one. The software architecture of such a robot is a key point for the efficiency of the communication between software modules achieving the tasks. According to the evolution of the task and to the behavior of the human, the system should react at the right level to provide the correct response in an acceptable time. In such a context the data exchanged between the software modules must be relevant and concise to make their processing fast enough. Also we propose to build the architecture around the concept of trajectory to take into account the time and synchronize the movements.

At the lower level, the robot must respond with reflex actions like reducing velocity or recoiling when the human approaches the robot. But for more important changes, the robot must replan its action and then switch from the previous trajectory to the new one satisfying HRI constraints. For more reactive tasks like the exchange of an object with a human, the robot must be able to compute and choose a good grasp and to compute a trajectory to reach and grasp the object in real time. These different robot behaviors must be integrated in the global robot architecture.

In this section, we present a quick review of the state of the art and then introduce our architecture to control a robot interacting with human.

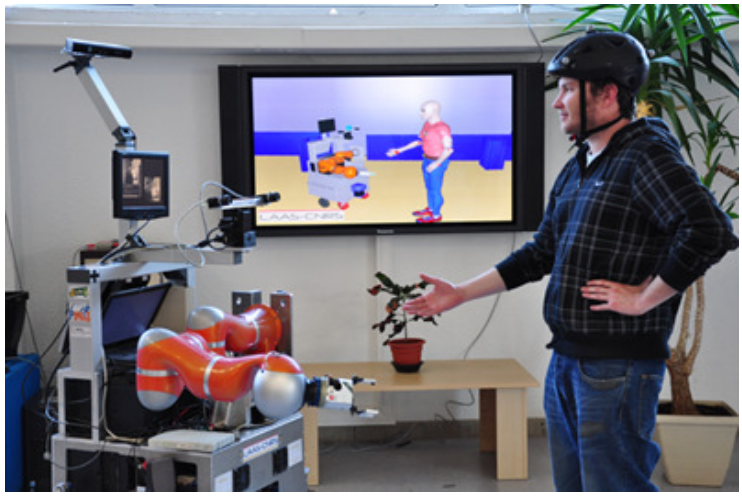


Figure 1: The Jido robot interacting with a human. The robot model of the scene is displayed on the wall-screen. The Kinect monitors the human kinematics and the human gaze is perceived using a motion capture system. Robot motions take the human into account at planning and execution level.

The introduction of robots that work among humans gives rise to new concepts and designs that were studied in recent years. The physical hardware as well as software components of the robot need to be designed by considering human's safety [1, 52]. Besides ensuring safety in robot hardware with compliant designs [39, 6, 68], the motions of the robot need to be "planned" and "executed" in a "human-aware" way by limiting the velocity at potential collision impact [33].

In [61] we have proposed a planning and control framework for synthesizing safe and socially acceptable robot motions. This framework was shown to generate human-aware motion for a static model of the human. In [45], we have extended the approach using a sampling-based "human-aware" path planner, which was based on a set of geometrical HRI constraints [60]. These constraints, taking as input the human kinematics and state, lead to safety, visibility and "arm comfort" costmaps defined over the robot configuration space. Sampling-based costmap planning techniques [40] were used to find good quality paths regarding the computed HRI cost criterion. Using users studies, Cakmak et al. [18, 17] have shown the importance of spatial and temporal pose of the robot for the exchange.

Executing motion bounded in jerk, acceleration and speed is also a way to produce human friendly robot motions. In [30], Flash and Hogan showed that the motion of the humans is by default limited in jerk and acceleration. Moreover human-robot object exchange studies [37] suggest that robot motion with minimum-jerk profile of the end-effector are preferred. In [11, 12], we have introduced a soft motion framework bounding the robot motion in jerk, acceleration and speed to ensure the human safety (speed) and comfort (jerk and acceleration).

The motion planning and execution frameworks of [61, 45] do not account for possible human motions during the trajectory execution. In motion planning literature, algorithms for dynamic environments have been introduced to take into account such changes in the robot workspace [9, 28, 69]. However, the human behaviors, which are considered in this chapter, do not lead to the same constraints as the moving obstacles taken into account by dynamic environments motion planning methods. In [9], a continuous set of homotopic paths is determined in which the initial path is deformed. Virtual forces are applied to the initial path by a control algorithm, the process can be viewed as an elastic band being stretched to gain optimality regarding clearance and length criteria. More recently in [28, 69], RRT-like algorithms have been introduced for motion planning with a limited time horizon well suited for dynamic environments. When executing the robot trajectory the human may come closer to the robot as shown in Figure 1, changing the safety and legibility constraints that have been taken into account by the path planner. Also, in handover situations the human may want to change the object transfer position (OTP), making the target configuration irrelevant to the task.

The HRI constraints [60] modeled as cost functions represent the amount of danger and how the human feels about a given robot configuration. Hence, as the danger of injuries increases and humans are frightened with high velocities, we propose to slow down the robot motion for high cost configurations by modifying on-line the timing-law without stopping the robot motion. This reactive scheme enables a safe and legible behavior according to human movements but it is not always efficient to account for the changes in the HRI costs. Hence we also propose to use the path perturbation variant of

[45] to optimize the executed solution regarding the current safety and comfort costs. In order to guarantee the jerk, acceleration and velocity bounds, we introduce an efficient way to replace a soft motion trajectory by a new one.

2.1 Architecture

From the high level decisional system that plans tasks and supervises execution to the actuators and sensors levels, the robot needs to compute many elements and disseminate data. As the robot must react at different levels, the architecture should irrigate software modules with the right flow of data, which is composed of sensors data, module results and decisions. The Figure 2 shows the architecture that we propose for tasks like pick and give or receive and place. This architecture and the associated modules can be improved and extended, but the properties described are sufficient to demonstrate the proposed functionalities. At the top level, task planner and supervision are intended to plan a task like "clear the table" or "pick this object and give it to this person" and then supervise the execution of the plan.

An important part of the data exchanged represents movements, which can be described by trajectories. As the human environment is changing, the robot must adapt its trajectories in real time. For example, if the human is approaching the robot, the velocity of the trajectory should be slowed. We present further in sect. 4.2 an interesting solution consisting in the use of series of cubic functions of time to represent trajectories.

The "path planner" uses RRT and T-RRT (see paragraph 4) to plan path as broken line for the whole robot in Cartesian or joint spaces. It is used to plan a first path and then to compute new paths in real time. For example if the robot is grasping an object from a human, and the "grasp planner" proposes a better grasp, this module computes a new path.

The "trajectory planner" transforms a broken line in soft motions satisfying the bounds in jerk, acceleration and velocity. It runs the "collision checker" and adapts jerk, acceleration and velocity limits from the costs associated to the position and behavior of the humans.

The human aware manipulation planner module (MHP) brings together the "path planner", the "grasp planner" and the "trajectory planner" to build a valid trajectory from the definition of the task and from the state of the robot provided by the SPARK module.

The SPARK module maintains a 3D model of the robot and of its environment (pose of objects, behavior of humans, position and posture of humans, visibility, etc.). This model is composed of known models and updated from data provided by the robot sensors.

The trajectory controller that monitors the execution of the trajectories is build on top of the controller provided by the robot manufacturer.

The "attentional system" uses sensors data that are preprocessed by the SPARK module to monitor and interpret the humans positions and behaviors. Given a model of the human behavior and the configuration of the robotic system, the attentional system can change the task that the robot is doing or the way the robot executes the task. For example, it can adapt the frequency of the clock that regulates the execution/control of

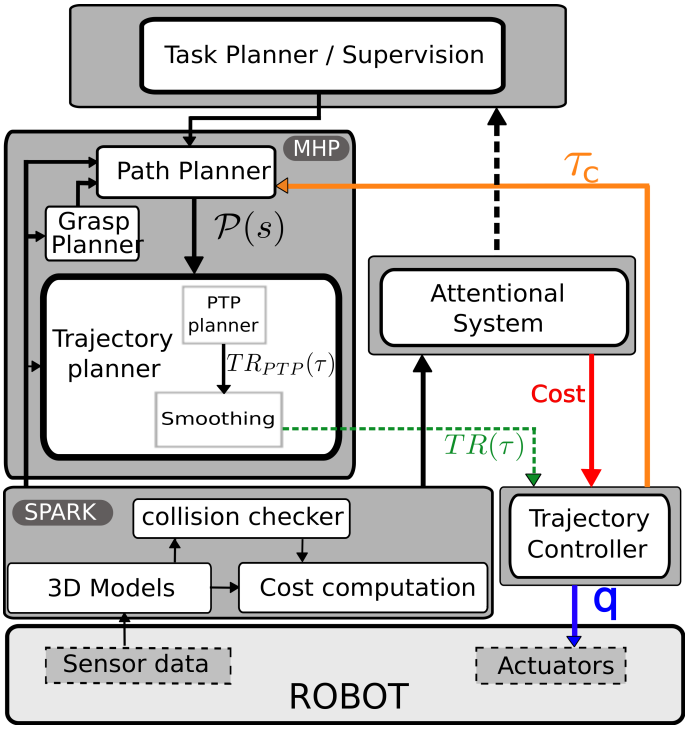


Figure 2: Architecture: from a robot movement defined by the task planner and supervision module and the state of the robot maintained by the SPARK element, the MHP module computes in real time a trajectory for the trajectory controller and the Attentional System monitors the execution.

the task. Finally, from the trajectory and state of the scene it can modify the trajectory, move the end position or adapt the cost to accelerate or decelerate the motion law. This possibilities are described in the Section 5.

In this architecture, the "grasp planner" has an important place as the choice of the initial grasp impacts all the task. For example, the position of the platform of the robot must be chosen so that the arm can achieve the grasp while avoiding obstacles. In the next section we detail a grasp planner developed in this context.

3 Grasp Planning

As the choice of a grasp to grab an object greatly determines the success of the task, we present here some aspects of the grasp planner module. For a complex object and simple tasks like pick and place or pick and give to a human, a lot of constraints have to be taken into account. But one essential point for human robot interaction (HRI) is the necessity of double grasp in many situations. Of course, both hands are required to lift a heavy object, but during the exchange of an object with a human the object is also

grasped by two hands. Sometimes, the robots needs to change the hand that holds an object and transitorily uses double grasp.

Grasp planning basically consists in finding a configuration for the hand(s) or end effector(s) that will allow picking up an object. If we consider a complete robotic platform, not only the grasp configuration is needed but also the configuration of the robot base and arm. To replace our work in the existing one, we give a brief overview of the state of the art concerning grasp planning in the next section.

3.1 Related Work

Most of the early grasp planning methods did not take into account finger nor arm kinematics and are often referred as contact-level techniques [51, 29, 26]. The contacts are regarded as freely-moving points with no link to any mechanical chain. Many grasp stability criteria have been introduced for this model of point/surface contact, the most common being certainly the force closure criterion [29, 5]. Force closure criterion is verified if a grasp can resist arbitrary force/torque perturbation exerted on the grasped object and is tested for a specific set of contacts (positions and normals). To integrate the notion of robustness of the grasp stability with respect to the contact positions, the concept of independent regions of contact has been introduced [51]. These regions are such that a grasp always verifies force closure as long as the contacts stay within the region. The computation of these regions has been solved for different object surfaces modelization (2D discrete surface [22], 2D polygonal surface [23], 3D polyhedral surface [55, 56]).

All these contact-level techniques were not very well-suited for real applications. Therefore, many new methods have appeared that integrate considerations on finger and/or arm kinematics.

Miller *et al.* [48] proposed to decompose the object into a set of primitives (spheres, cylinders, cones and boxes). With each primitive is associated a pregrasp configuration of the hand. A set of parameters is sampled in order to test the different possible approaches of the hand, then, for each approach, the fingers are closed on the object until collision. The quality of the obtained grasp is then computed according to the measure described in [29].

The idea of object decomposition was widely used and is still the base of many grasp planners. It offers a heuristic to reduce the possible relative palm/object poses to test. In [32], the authors decompose the object model into a superquadric *decomposition tree* employing a nonlinear fitting technique. Grasps are then planned for each superquadric with a heuristic approach close to the one in [48]. The grasps are then simulated on the original object model using the GraspIt! dynamics simulator [47], to sort them by quality.

Huebner *et al.* [38] proposed a technique to build a hierarchy of minimum volume bounding boxes from 3D data points of the object envelop. This method offers a interesting robustness with respect to the quality of the object 3D model, acquired from sensors (here laser scan).

In [36], the object is decomposed into a set of boxes called *OCP* (Object Convex Polygon). Each box of the OCP is compared to a *GRC* (Grasping Rectangular Convex), which gives an estimation of the maximum size of the object that the hand can grasp.

Different GRCs are defined corresponding to different grasping styles. Xue *et al.* [66] presented a method to optimize the quality of the grasp while taking into account the kinematics of the fingers during the optimization phase. They use a swept volume precomputation associated with a continuous collision detection technique to compute, for a given hand/object relative pose, all the possible contacts of each finger on the object surface. After obtaining an initial grasp provided by the GraspIt! software [47], they locally optimize the quality of the grasp in the finger configuration space.

Some works gave more focus on arm and/or robot base inverse kinematics issues. Berenson *et al.* [4] are interested in finding grasp configurations in cluttered environments, for a given robot base position in the object range. From different object approaches, the authors precompute a set of grasps, all verifying the force closure property. Instead of trying to solve the arm inverse kinematics and checking for collisions for each grasp of the set in an arbitrary order, the authors propose to compute a *grasp scoring* function for each grasp. The function is used to evaluate the grasps that are more likely to succeed the inverse kinematics and collision tests and is based upon a force closure score, a relative object-robot position score and an environment clearance score.

The authors of [25] focused on path planning for the robot base (or body) and arm and presented a planning algorithm called *BiSpace*. Like in [4], they first compute a set of grasp configurations for the hand alone. Once one or more collision free configurations for the hand are found, they become the start nodes of several RRTs (Rapidly Random-exploring Tree [44]) that will explore the hand workspace while another RRT is grown from the robot base start configuration, that explores the robot configuration space.

Some recent works were inspired by results in neuroscience [57, 65] which have shown that humans mainly realize grasping movements that are restricted in a configuration space of highly reduced dimensionality. From a large data set of human pregrasp configurations, Santello *et al.* [57] performed a principal component analysis revealing that the first two principal components account for more than 80% of the variance. Ciocarlie *et al.* [20] called the components *eigengrasps* and use them as a base to represent the reduced configuration space of the hand. They also add the six DOF's of the wrist pose. Then, they use a simulated annealing based optimization method, in eigengrasp space, to find the best grasp according to an energy function. The energy function takes into account two parameters. First, the distance between specified points on the hand and the object surface. Secondly, a quality metric based on the one in [29].

A frequent difficulty associated with grasp planning concerns the 3D model reconstruction of the object to be grasped. This reconstruction is not an easy task and the resulting model can be very noisy. In order to avoid the need for 3D-model reconstruction, Saxena *et al.* [58] proposed a method to find good grasps of objects being seen for the first time, that does not require such a model. This method is based on a learning approach that uses image features to predict good points where to grasp the object. These features are based on edges, textures and colors. A set of generated synthetic images of various objects is used to learn the feature values of region labeled as grasping points. For a novel object, a probabilistic model of the grasping point features is used to find grasping points in the image. A triangulation is then performed that uses images from different points of view to find the region where to grasp the object in 3D

space.

3.2 The Grasp Planner

As explained above, for HRI, grasp planning has several uses and is not only devoted to basic pick-and-place tasks. In particular, in a planning point of view, the context is very important in order to choose a valid grasp. Therefore, the proposed approach does not rely exclusively on a heuristic that can introduce a bias on how the object is grasped. Our objective is to build a grasp list to capture the variety of the possible grasps. It will then allow finding a grasp, even in a cluttered environment, for an object with a complex shape. In the following, we illustrate the method with the Schunk Anthropomorphic Hand (SAH) depicted on Fig. 3 as it is the one used in our laboratory. It has four fingers. Each finger, except for the thumb, has four joints. Only the three first joints are actuated, the last one being coupled with the third one. The thumb has an additional actuated joint to place it in opposition to the other fingers. The method however applies to other hand kinematic structures, after some small numeric adaptations.

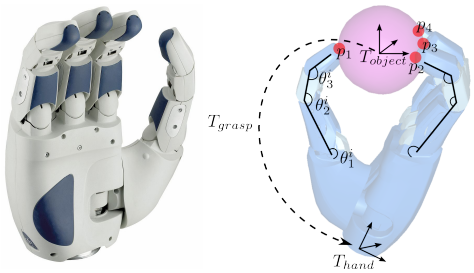


Figure 3: Left: The Schunk Anthropomorphic Hand used to illustrate our grasp planning method. Right: A grasp is defined by a transform matrix T_{grasp}, the finger joint parameters of each finger i $(\theta_1^i, \theta_2^i, \ldots)$ and a set of contact points $(p_1, p_2, \ldots)$.

A single grasp is defined for a specific hand type and for a specific object. The object model is supposed to be a triangle mesh: A set (array) of vertices (three coordinates) and a set of triangles (three indices in the vertex array). It is assumed to be a minimum consistent *i.e.* has no duplicate or isolated vertices nor degenerate triangles.

3.2.1 Grasp Definition

In the following, we define a grasp by (See Fig. 3):

- A transform T_{grasp} between the object and the hand frame.
- A set of finger joint parameters $(\theta_1^i, \theta_2^i, \ldots)$ where i is the ID of the finger.
- A set of contact points $(p_1, p_2, \ldots)$ that can be deduced from the two previous items.

A contact contains the following information:

- Position: both a 3D vector and a set (triangle index + barycentric coordinates) to store the position.
- Normal: the plane normal of the triangle the contact belongs to.
- Coulomb friction: used further to compute the grasp stability.
- Finger ID: to store which finger is responsible of the contact.
- Curvature: it is interpolated from the curvature of the vertices of the triangles.

As the main concern of the grasp planner is motion planning, it is not possible to rely on the computation of an only grasp or to compute grasps according to a heuristic that could introduce a bias on the choice of the grasp. It is preferable to compute a grasp list that aims to reflect the best the variety of all possible grasps of the object. Our algorithm applies the following steps that will be detailed further:

- Build a set of grasp frame samples.
- Compute a list of grasps from the set of grasp frames.
- Perform a stability filter step.
- Compute a quality score for each grasp.

3.2.2 Grasp Frame Sampling

For manipulation planning, it is important to avoid biasing the possible approach of the hand when we compute the grasp. Therefore, we choose to sample the possible grasp frames uniformly. This is done by the mean of a grid. We have chosen, for our hand, a grasp frame that is centered on the intersection of the finger workspaces so that it is roughly centered where the contacts may occur. We set as an input the number of positions and the number of orientations, each couple position-orientation defining a frame. The positions are uniformly sampled in the object axis-aligned bounding box with a step computed to fit the desired number of position samples. The orientations are computed with an incremental grid like the one in [67]. For each grasp frame, a set of grasps will be computed.

3.2.3 Grasp List Computation

As the proposed grasp planning method does not restrict the possible hand poses or surfaces of contact on the object, it requires a lot of computation. Therefore, we have to introduce some data structures to reduce the computation times. Except for collision test, the most expensive computation is the finger inverse kinematics. One has to be able to know the fastest possible if, for a specified hand pose (relative to the object), a finger can establish a contact on the object surface and, if it is the case, where. The contacts can only occur in the intersection of the finger workspace and the object surface. For each finger, it is consequently crucial to find this intersection or at least an approximation. We use two data structures to model the object surface and finger workspaces.

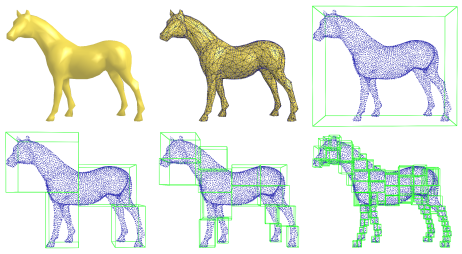


Figure 4: The object mesh is uniformly sampled with a point set (top images). The point set is then partitioned using a kind of kd-tree (bottom images).

3.2.4 Object Surface Model

We propose to approximate the object surface with a contact point set, keeping trace of where it is on the object mesh to be able to get some local information (surface normal and curvature) later. The set is obtained by a uniform sampling of the object surface. The sampling step magnitude is chosen from the fingertip radius. A space-partitioning tree is built upon the point set in order to have a hierarchical space-partitioning of the points (Fig. 4). It is similar to a kd-tree. Starting from the original set of points, we compute the minimal axis-aligned box containing all the points. Such a box is usually referred as Axis-Aligned Bounding Box or AABB. This first AABB is the tree root. The root AABB is then splitted in two along its larger dimension. This leads to two new nodes, children of the root, containing each a subset of the original point set. The splitting process is then recursively applied to each new node of the tree. The process ends when a node AABB contains only one point.

We then need to find the intersection of each finger workspace with the object surface tree. So we introduce another data structure to approximate the finger workspace and compute this intersection quickly.

3.2.5 Finger Workspace Model

As spheres are invariant in rotation, they are interesting to build an approximation of the finger workspace. Starting from a grid sampling of the finger workspace (Fig. 5), we incrementally build a set of spheres fitting strictly inside the workspace. First, points of the grid are marked as being boundary points (on the workspace envelope) or inner points (strictly inside the workspace volume). For each inner point, the smallest distance to the boundary points is computed, referred as *dmin*. The inner point having the biggest *dmin* is the center of the first sphere S_1, of radius *dmin*. For all the inner points that are not inside S_1, a new *dmin* is computed, that is the minimum of the old *dmin* and the minimal distance to S_1. The point that has the biggest *dmin* is the center of the second sphere S_2, of radius *dmin*. This process is repeated until we have reached the maximal desired sphere number or the last computed sphere has a radius less than a specified threshold. We keep the ordering of the construction so that the sphere hierarchy starts from the biggest ones, corresponding to workspace parts that

are the farthest to the finger joint bounds. These bounds were first slightly reduced (Fig. 5) in order to eliminate configurations where the fingers are almost completely stretched.

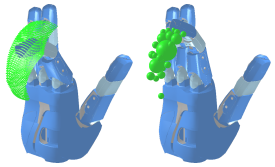


Figure 5: The finger workspace, discretized with a grid (forefinger workspace, left image). The grid is converted to a volumetric approximation as a set of spheres (right image).

Once we have both the contacts tree and the workspace sphere hierarchy, it is very fast and easy to determine the intersection of the two sets and so the contact points.

3.2.6 Intersection Between Object Surface and Finger Workspace

All the operations that have to be performed are sphere-box intersection tests. The intersection is tested from the biggest to the smallest sphere, guaranteeing that the *best* parts of the workspace will be tested first, *i.e.* the ones farthest to singularities due to the joint bounds. Starting from the tree root, we test if there is a non-null intersection between a AABB-node and the sphere. If not, we stop exploring this branch, otherwise we test the sphere against the two node children, until we arrive to a leaf node, *i.e.* a single point. We then just have to test if the point is included in the sphere volume. Fig. 6 shows the contact point candidates for two different grasp frames with the same object. At this stage, we just know that the points will pass the finger inverse kinematics test. No collision tests have been performed yet. For a given grasp frame, the grasp is

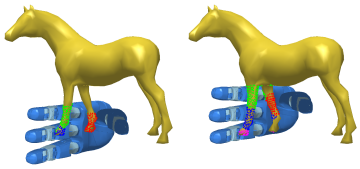


Figure 6: The potential finger contacts, drawn in red, green, blue and magenta for the thumb, forefinger, middle finger and ring finger respectively. On the left image, no contact can be found for the ring finger because of its limited workspace.

computed finger by finger, that means that, if we have the contact and configurations of the fingers 1 to $i-1$, we search a contact point for finger i and test collision only with the fingers 1 to i as the other finger configurations are not yet known. We start from the thumb as no stable grasp can be obtained without it. If a finger can not establish a

contact, it is left in a *rest* (stretched) configuration. If we have three contacts or more, we can proceed to the stability test. Note that, at this stage, we have a collision-free grasp, *i.e.* no collision between the hand and the object and do not yet consider collision with the environment or the robot arms or body.

3.2.7 Stability Filter and Quality Score

The stability test is based on a *point contact with friction* model, that explains why at least three contacts are required. From the contact positions and normals, we compute a stability score. It is based on a force closure test and stability criterion [7]. All the grasps that do not verify force-closure are rejected. We also compute and add a second score that is the distance to the mass center of the object. The stability score is not sufficient to discriminate good grasps so we build a more general quality score.

Several aspects can be taken into account to compute a grasp quality measure [63]. A trade-off is often chosen with a score that is a weighted sum of several measures. We chose to combine the previous stability criterion with two other criteria: A finger force ellipsoid major axis score and a contact curvature score. The idea behind the first one is that it is preferable to favor contact such that the contact normal is in a direction close to the direction of the major axis of the force ellipsoid, corresponding to the better force transmission ratio. Fig. 7 shows the force ellipsoids computed for a configuration of the SA Hand.

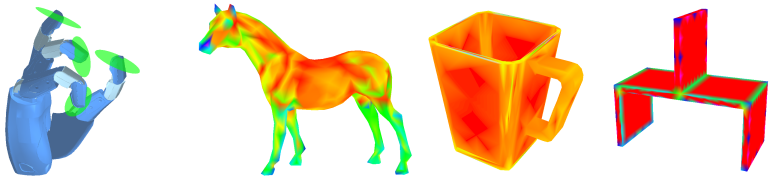


Figure 7: Left: the finger force ellipsoids must fit the contact normal to ensure a good grasp. Right: the mean curvature of the object surface is used as a quality criterion on the contact position. Surface color varies from red (low curvature) to blue (high curvature), through green.

The curvature score is used to favor contacts where the mean curvature of the object surface is low. In real situation, it will reduce the impact of a misplaced contact as the contact normal will be susceptible to smaller change in a low curvature area than in a high curvature one. Fig. 7 shows, on some objects, how low curvature areas are preferable to establish contacts. Curvature is computed for each vertex and then interpolated for each point on the surface from its barycentric coordinates. The curvature is then normalized to be always included in $[0; 1]$.

3.2.8 Double Grasp Planning

A double grasp is a grasp involving both hands. It is computed from two single grasp lists L_1 and L_2, obtained for each hand. The model for double grasp simply derives

from the single grasp model: it is defined by a valid grasp for each hand and the two associated quality.

Each single grasp pair sg_1 and sg_2, belonging to L_1 and L_2 respectively, is tested. All colliding pairs are rejected. The list can be filtered once we have more information about the environment or task to realize. For instance, if the task is to pick up an object with one hand, give it to the other hand before placing it on a support, we can remove all the grasps that lead to a collision with the environment for the given initial and final object poses. For instance, all the grasps that take the object from *below* will be removed as they lead to a collision between the object support (*e.g.* a table) and the hand. For each double grasp, a score is then computed based on two scores: The quality of each single grasp and an inverse kinematics (IK) score.

- The minimum of sg_1 and sg_2 quality is used as a stability score for the double grasp.
- An IK score is computed for sg_1 and sg_2. It is based on how *natural* is the way to grasp the object in its start and goal configuration using sg_1 and sg_2. The score is a distance of the arm configuration to a predefined rest configuration. For the double grasp, we take the minimum of the IK scores of sg_1 and sg_2.

After normalizing these two scores separately for all the computed double grasps, we sum them for each double grasp to obtain its score.

Fig. 8 shows a double grasp computed with our algorithm.

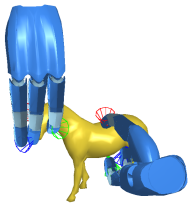


Figure 8: An example of double grasp computed for right and left SAHs with friction cones displayed in colors.

3.2.9 Double Grasp for Object Transfer

First, a double grasp list must be computed for the object of interest. This list is computed for a hand pair composed of the robot hand and a human hand (his/her right hand *a priori*). The human hand model is of course a simplification as our modelization only deals with rigid bodies. We use the SAH as it is already available, but with a scaled kinematic structure to approximate the human hand.

Let note a double grasp of the list $dg = (sg_r, sg_h)$ where sg_r is a grasp of the robot hand while sg_h is a grasp of the human hand. For a given object, placed on a support in a particular environment, we remove, from the previously computed double grasp list, all the double grasps such that sg_r does not allow the robot to grasp the object. From

each remaining double grasp, knowing the positions of both human and robot, we can deduce how to hand over the object to the human as the grasp gives the direction of the human wrist to grasp the object. The robot still has to choose a double grasp from the list and an exchange pose for the object. The choice of the double grasp is based on the notion of *intention legibility*. It must be easily interpreted as an object transmission. The best double grasps appear then to be the ones where the wrist directions of human and robot are opposed. The choice of the exchange pose is based on the notion of comfort. It must allow the human to grasp the object with a comfortable wrist/arm direction, *i.e.* directed from the human position to the object position (Fig. 9).

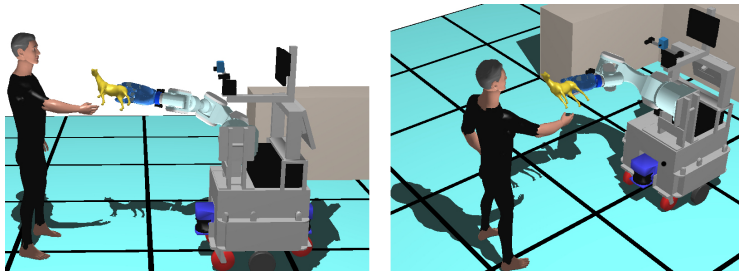


Figure 9: A good object transmission must be easily interpretable from the human point of view and must not require an uncomfortable arm movement for the human.

3.2.10 Dual Arm Manipulation

Dual hand/arm grasps are at least required in two situations, when the object to carry is too heavy to be carried with only one hand and when the robot has to transfer the object from one hand to the other to take advantage of the workspaces of the two arms. In this re-grasping case, a first solution consists in placing the object on a support and then picking it up again with the other hand. But a better solution is to use the second hand to realize a temporary dual-handed grasp before removing the first hand.

For a given manipulation task, the robot will start with a one-handed grasp g_i and end with a one-handed grasp g_f. The regrasping task will be achieved with the help of a double grasp obtained by combining g_i and g_f. As the hands must not collide during the regrasping task, g_i and g_f must be chosen appropriately. Grasps that were ideal in the case of single-handed manipulation are generally no more usable for dual-handed manipulation. Indeed, for stability reason it is preferable to use contacts that are close to the object center of mass. This leads to configurations where the hand is centered on the object, that do not let enough room to take the object with the other hand. When the robot uses dual-handed grasps, it will modify the initial and final single-handed grasps in order to take the object on its extremities. Such an example is depicted in Fig. 10 where the DLR's robot Justin [54] manipulates a horse statuette. The best grasps, in term of stability, are on the body of the horse. However, it is not possible to place two hands on this part of the object. Consequently, the robot has to choose to grasp the extremities of the object (leg and head on the example).

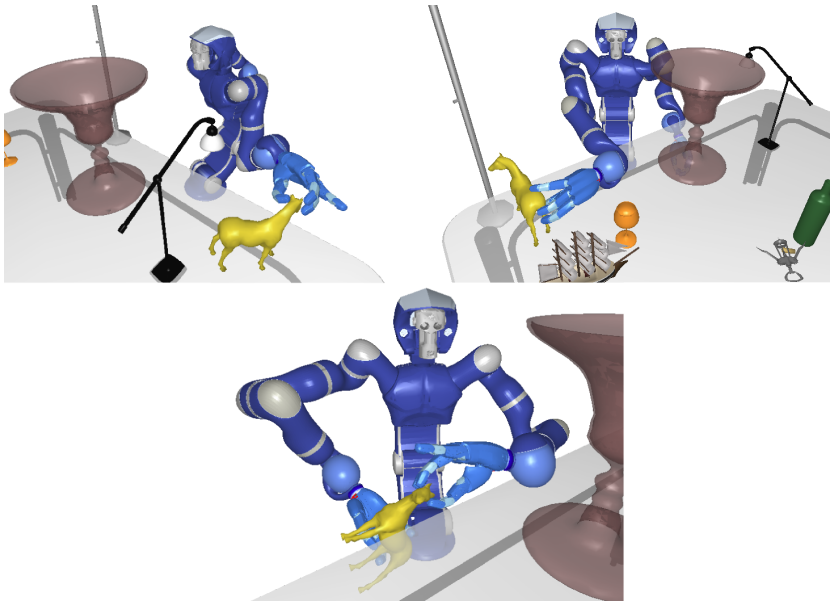


Figure 10: If the robot (DLR, [54]) has to realize a regrasping task, it must select initial and final grasps that let enough room to perform a dual-handed grasp.

It is also possible to perform regrasping to just modify the grasp of one hand. Let us suppose the robot holds the object with the right hand and with a grasp g_{right}. The robot also needs to change the grasp. It can take the object with its left hand and a grasp g_{left}, release the first grasp, possibly re-orient the object, and grasp the object with right hand again but with a different grasp g'_{right}. Two dual-handed grasps will thus be required: $[g_{right}, g_{left}]$ and $[g'_{right}, g_{left}]$. The grasp selection uses the same principle as above but is more combinatorially complex.

This technique has been implemented for the robot Justin[1] equipped with two SAHs, within our simulator, (Fig. 10) to plan pick-and-place tasks that require regrasping.

We have presented a grasp planner for single and double grasp that allows choosing a grasp in real time according to the behavior of the human. As we have seen, grasp planning is complex and greatly determines the succes of a manipulation task. In particular, the choice of a grasp compatible with the whole task is crucial. In the next section, we introduce how to plan and adapt a displacement after the move is defined in an interactive context.

[1] Justin is a robot of DLR that gracefully made the model available for LAAS-CNRS

4 Motion Planning

The HRI introduces real challenges for the motion planning problem. While motion planning is not yet largely used in industry where most robots are still programmed by learning, for HRI we need to plan and adapt in real time motions that take into account human movements and behaviors. Traditional motion planners plan only a path that a controller executes at constant velocity. To take into account human motion, we propose here to plan trajectories that satisfy the HRI constraints: **safety and comfort**.

From an elementary task like "pick an object", "place an object" or "give an object", the motion planner must precise initial and final conditions for each move, plan a trajectory and then adapt the trajectory in real time. For example, to plan the first movement of a pick and give task, the planner must firstly choose an initial grasp that takes into account the double-grasp needed to give the object and then adapt the movement to the human behavior.

In this section, we present a first skeleton of a planner for human-robot interactive motions.

4.1 Planning The Path

The first step of the motion planning is the computation of the path $\mathscr{P}$. In our case, the motion planner is based on the planner initially proposed by Sisbot [60]. The motion planner takes explicitly into account the constraints of HRI to synthesize navigation movements (movement of the platform) and handling (fixed robot platform). HRI constraints are for example the human-robot distance or field of view of the human. This planner is based on case studies in HRI [42] and on existing theories on the proxemic behavior of humans [34]. The HRI constraints are represented by cost functions based respectively on the posture of the human, his/her field of view and his accessibility. These costs are represented by costs maps defined in the working space of the robot. In [60], to solve a manipulation task like passing an object to a human, the path of the object is first planned using grids methods defined in the workspace. Then the path of the robot is planned from the inverse kinematic of the robot. However, as the path of the object is defined by the first step of the method, the original planner is not efficient in cluttered environment. We use the extension proposed by Mainprice [45]. This extension consists in extending the capabilities of the planner through the use of planning algorithms based on random sampling to compute the moves taking into account human in cluttered environments.

4.1.1 Random Path Planner

When the robot shares the workspace with humans, the path planner must take into account the costs of HRI constraints. We perform this planning with the T-RRT method [40] which takes advantage of the performance of two methods. First, it benefits from the exploratory strength of RRT-like planners [43] resulting from their expansion bias toward large Voronoi regions of the space. Additionally, it integrates features of stochastic optimization methods, which apply transition tests to accept or reject potential states. It makes the search follow valleys and saddle points of the cost-space

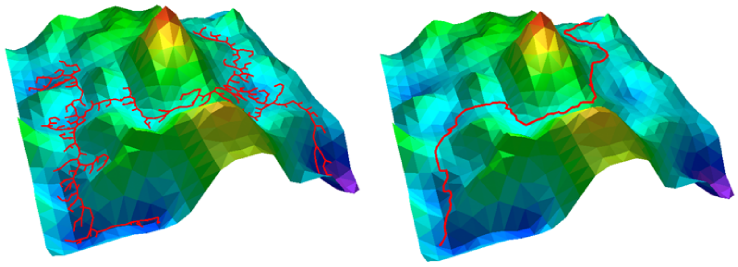


Figure 11: T-RRT constructed on a 2D costmap (left). The transition test favors the exploration of low-cost regions, resulting in good-quality paths (right).

in order to compute low-cost solution paths (Fig. 11). This planning process leads to solution paths with low value of integral cost regarding the costmap landscape induced by the cost function.

In a smoothing stage, we employ a combination of the shortcut method [3] and of the path perturbation variant described in [45]. In the latter method, a path $\mathscr{P}(s)$ (with $s \in \mathbb{R}^+$) is iteratively deformed by moving a configuration $q_{perturb}$ randomly selected on the path in a direction determined by a random sample q_{rand}. This process creates a deviation from the current path, The new segment replaces the current segment if it has a lower cost. Collision checking and kinematic constraints verification are performed after cost comparison because of the longer computing time.

The path $\mathscr{P}(s)$ computed with the human-aware path planner consists of a set of via points that correspond to robot configurations. Via points are connected by localpaths (straight line segments). Additional via points can be inserted along long path segments to enable the path to be better deformed by the path perturbation method. Thus each localpath is cut into a set of smaller localpaths of maximal length l_{max}.

4.1.2 Taking into Account Geometric Constraints

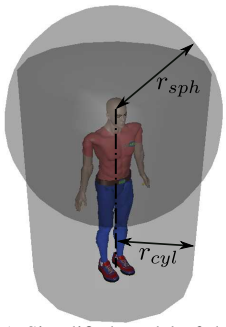
We use costmaps to model the HRI. These costs are important when the configuration of the robot is not safe or comfortable for the human. We retain three constraints:

- **Safety constraint** (Fig. 12(a)): This constraint ensures the safety of interaction by monitoring the distance between the robot and the human. The human is modeled by approximating the bounding volume of his/her body (regardless of the geometry of the arm). To reduce the risk of collision between the human and the robot, this safety constraint keeps the robot away from the head and body. The distance to be considered is the minimum distance between the robot (all parts of the robot are taken into account) and the simplified model of human (cylinder + sphere). When this distance is small, the cost is high and conversely when the distance increases the cost decreases to a minimum threshold after which it becomes null.

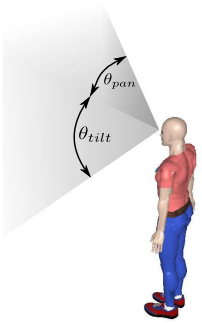
- **Visibility constraint** (Fig. 12(b)): This constraint aims at limiting the effect of surprise that may experience a human while the robot moves in the workspace. A

human feels less surprised if the robot remains visible and the interaction is safer and more comfortable. Thus, each point of the workspace has a cost proportional to the angle between the gaze of the human and his/her position in the Cartesian space;

- **Constraint of "comfort of the human's arm"**: The third constraint is taken into account for object exchange tasks between robot and human. It allows determining an object transfer point (OTP) in the workspace. This constraint is also taken into account during the path planning of the exchange task to facilitate the exchange of the object at any time during the move. For this, the robot must reason on the kinematic and the accessibility capabilities of the human. The assumed reachable volume of the human can be pre-computed using generalized inverse kinematics. For each point inside the reachable volume of the human, the determined configuration of the torso remains as close as possible to a given resting position. Collision detection with the environment is then used to validate these postures. At each valid position, a comfort cost is assigned through a predictive model for human posture introduced in [46].



(a) Simplified model of the human for the safety cost.



(b) Visibility model of a human.

Figure 12: Cost models of safety and visibility.

Each constraint is represented by a three-dimensional cost map, these basic costmaps are then combined with a weighted sum:

$$cost(h,q) = \sum_{i=1}^{3} w_i c_i(h,q) \tag{1}$$

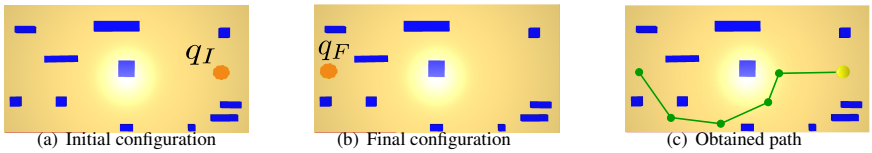
where w_i are the weights, h the human posture, q the robot configuration and c_i, the costs.

In the current implementation, the weights are empirically defined and cost functions are evaluated "on the fly" during planning.

4.1.3 Path Planning

According to the presence or the absence of the human in the scene, the path planning is performed using T-RRT if human is present or RRT if not.

In both cases, the path resulting $\mathscr{P}(s)$ ($s \in \mathbb{R}^+$) is composed of a set of robot configurations (nodes) connected by straight lines (edges). Consider the example of a two-dimensional system solved by the method RRT. the Figures 13(a) and 13(b) respectively represent the initial and final positions of the yellow puck. The path obtained is shown in Fig. 13(c) (green lines connecting the spheres). The spheres represent the intermediate configurations of the path.



(a) Initial configuration (b) Final configuration (c) Obtained path

Figure 13: Example of a the planning of a path in a 2 dimension space.

4.2 From the Path to the Trajectory

We propose to generate a trajectory from a path using the soft motion trajectory planner designed by Broquère [12, 11, 10].

4.2.1 Trajectory Model

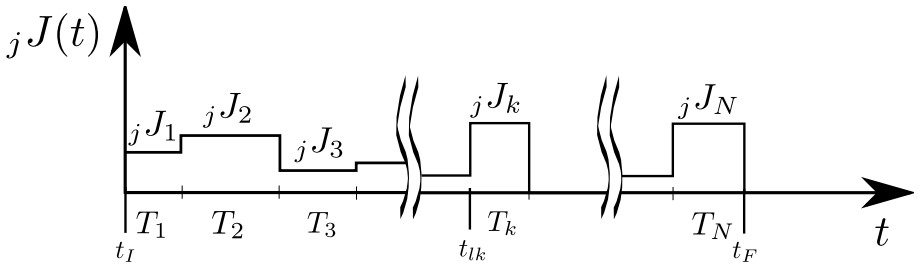


Figure 14: The jerk evolution for the j axis of the TR(t) trajectory.

A trajectory TR(t) is represented by a combination of n series of cubic polynomial curves. The use of polynomial cubic defined by the Soft Motion Trajectory Planner provides a solution in the context of HRI where the task introduces numerous constraints. From the trajectory generation point of view the safety constraint is ensured by bounding the velocity and the comfort constraint by bounding the jerk and the acceleration.

The trajectory ${}_j\mathrm{TR}(t)$ corresponds to the evolution of the j axis and is composed of N cubic polynomial segments (curves) (Fig. 14). We consider that each axis has the same number of segments since they can be divided.

Functions ${}_jJ_k(t)$, ${}_jA_k(t)$, ${}_jV_k(t)$, ${}_jX_k(t)$ respectively represent the jerk, acceleration and velocity evolution over the k segment for the j axis. T_i is the initial time of the trajectory and T_F the final one.

A segment is defined by the Eq. (2) and depends on its duration T_k and on five parameters:

- the initial time t_{lk} with $t_{lk} = t_I + \sum_{i=1}^{k-1} T_i$,
- the initial conditions (3 parameters: ${}_jA_k(t_{lk})$, ${}_jV_k(t_{lk})$, ${}_jX_k(t_{lk})$),
- the jerk value ${}_jJ_k$

$\forall t \in [t_{lk}, t_{lk} + T_k]$:

$$ {}_jX_k(t) = \frac{{}_jJ_k}{6}(t - t_{lk})^3 + \frac{{}_jA_k(t_{lk})}{2}(t - t_{lk})^2 + {}_jV_k(t_{lk})(t - t_{lk}) + {}_jX_k(t_{lk}) \tag{2} $$

where ${}_jJ_k$, ${}_jA_k(t_{lk})$, ${}_jV_k(t_{lk})$, ${}_jX_k(t_{lk})$ and t_{lk} are constant $\in \mathbb{R}$.
The initial conditions of the trajectory ${}_j\mathrm{TR}(t)$ are:

$$ \begin{aligned} {}_jA_1(t_I) &= {}_jA_I \\ {}_jV_1(t_I) &= {}_jV_I \\ {}_jX_1(t_I) &= {}_jX_I \end{aligned} \tag{3} $$

and the final conditions:

$$ \begin{aligned} {}_jA_N(t_F) &= {}_jA_F \\ {}_jV_N(t_F) &= {}_jV_F \\ {}_jX_N(t_F) &= {}_jX_F \end{aligned} \tag{4} $$

where $t_F - t_I = \sum_{i=1}^{N} T_i$.

The multidimensional trajectory is then a composition of trajectories as:

$$ \mathrm{TR}(t) = \left[{}_1TR(t)\ {}_2TR(t)\ \ldots\ {}_nTR(t) \right]^T \tag{5} $$

where n is the number of axis.

From the N couples $({}_jJ_k, T_k)$ and the initial conditions (3) of the trajectory ${}_j\mathrm{TR}(t)$ we can compute the kinematic state along the j axis at a given time with (6), (7) and (8). In order to simplify the notation, the j index representing the axis will be omitted.

$\forall t \in [t_{lk}, t_{lk} + T_k]$, with $t_I = 0$:

$$A_k(t) = J_k(t - \sum_{i=1}^{k-1} T_i) + \sum_{i=1}^{k-1} J_i T_i + A_I \tag{6}$$

$$V_k(t) = \frac{J_k}{2}\left(t - \sum_{i=1}^{k-1} T_i\right)^2 + \sum_{i=1}^{k-1} J_i T_i \left(t - \sum_{j=1}^{i} T_j\right) + \sum_{i=1}^{k-1} \frac{J_i T_i^2}{2} + A_I t + V_I \tag{7}$$

$$X_k(t) = \frac{J_k}{6}\left(t - \sum_{i=1}^{k-1} T_i\right)^3 + \sum_{i=1}^{k-1} \frac{J_i T_i}{2}\left(t - \sum_{j=1}^{i} T_j\right)^2 + \sum_{i=1}^{k-1} \frac{J_i T_i^2}{2}\left(t - \sum_{j=1}^{i} T_j\right) + \sum_{i=1}^{k-1} \frac{J_i T_i^3}{6} + \frac{A_I}{2} t^2 + V_I t + X_I \tag{8}$$

4.2.2 The Kinematic Constraints

The trajectory generation method is based on constraints satisfaction (velocity, acceleration and jerk). Each constraint is supposed constant along the planned motion. In the multidimensional case, each axis can have different constraints. We also suppose that the constraints are symmetrical:

$$\begin{aligned} {}_jJ_{min} &= -{}_jJ_{max} \\ {}_jA_{min} &= -{}_jA_{max} \\ {}_jV_{min} &= -{}_jV_{max}. \end{aligned} \tag{9}$$

Hence, the jerk, acceleration and velocity must respect:

$$\begin{aligned} |{}_jJ(t)| &\leq {}_jJ_{max} \\ |{}_jA(t)| &\leq {}_jA_{max} \\ |{}_jV(t)| &\leq {}_jV_{max}. \end{aligned} \tag{10}$$

4.3 Basic Concepts of the Trajectory Generation

This section describes breifly the core of the trajectory generator bounding the jerk, the acceleration and the velocity. Details can be found in [12, 11, 10]. The three introduced methods do not use optimization steps, they are designed to be used on-line in a control loop to modify the trajectory and, for example, track and catch an object handled by the human.

4.3.1 The Canonical Case: the Kinematically Constrained Point-to-Point Motion

In the basic case a motion between two points where initial and final kinematic conditions are null, the Figure 15 represents the optimal point-to-point motion (according to the imposed kinematic constraints). This point-to-point motion is composed of seven segments of cubic polynomial functions at most [12].

In the multidimensional case each axis has also seven cubic polynomial segments at most. Computation details can be found in [10].

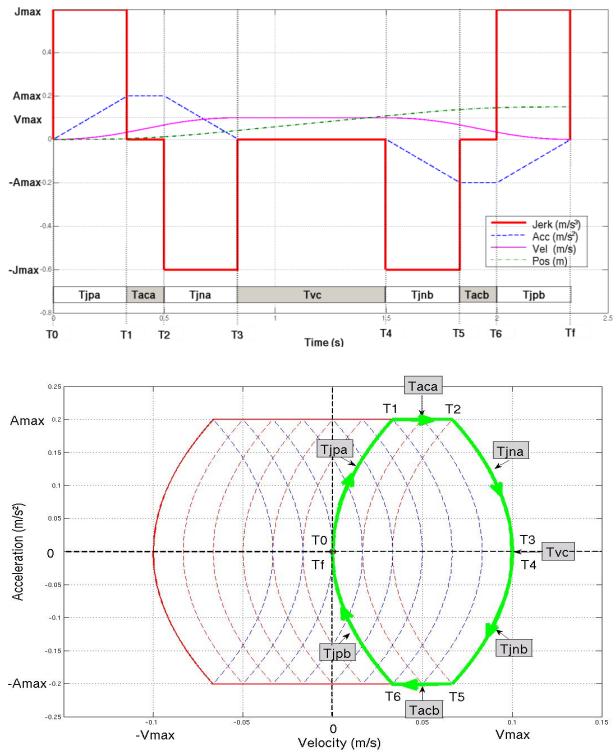


Figure 15: Jerk, acceleration, speed and position curves and motion in the acceleration-velocity frame for a single axis.

4.3.2 The Minimal Time Motion Between Two Non-null Kinematic Conditions

From the canonical point-to-point case we extend the monodimensional algorithm to compute minimal time motion between two non-null kinematic states (non-null acceleration and velocity). An overview of this algorithm is presented in [12] and the details in [10]. This kind of motion is composed of a set of elementary motions saturated in jerk, acceleration or velocity. The number of elementary motions is also seven at most. For the multidimensional case, we propose in [10] a solution to synchronise the axis motions.

4.3.3 The Time Imposed Motion Between Two Non-null Kinematic Conditions: the *3-Segment Method*

The method for computing a motion with an imposed duration was previouly presented in [11]. This method does not bounds the jerk, acceleration nor velocity. It uses three cubic polynomial curves to define such a motion. This simple definition provides a solution to compute analytically the motion.

4.3.4 Smoothing an Input Function

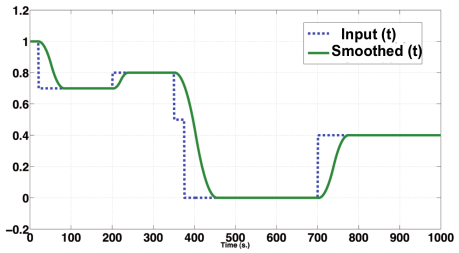


Figure 16: Example of the smoothing of a set function.

We use the method proposed in [12] to compute online a smooth movement from an input defined by acceleration and velocity. At each update of the set function, a move is computed from the current state of the system. This move is bounded by the kinematic constraints (J_{max}, A_{max} and V_{max}). Under this kinematic constraints, the minimal time motion is defined by the critical movement associated to the critical length dc [12].

Thus, in order to allow a mono-dimensional system to reach its set value in minimal time, the critical movement is computed at each iteration.

An example of a smoothed signal is plotted in the Fig. 16. The blue dotted curve is the input and the green curve is the smoothed velocity. The method acts like a filter for the acceleration.

4.4 Trajectory Generation

The trajectory generation is based on the three main methods introduced in the previous section. The input is the path $\mathscr{P}$ computed by the path planner (Sect. 4.1.1).

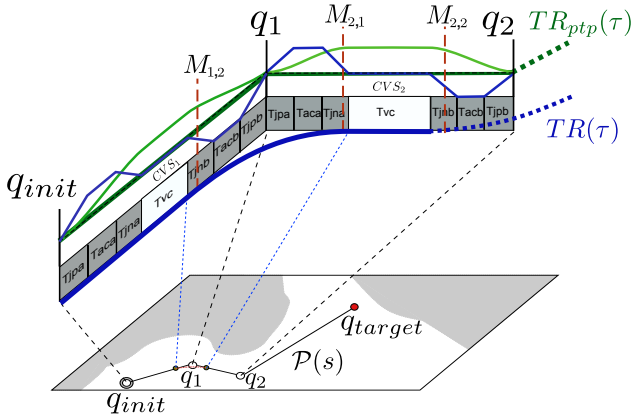


Figure 17: From the path $\mathscr{P}$ to the smoothed trajectory TR.

The first step is to calculate a trajectory passing through all the nodes of the path $\mathscr{P}$. This trajectory, which we call TR_{ptp} consists of point-to-point movement (Sect. 4.3.1) and therefore includes stop motions at each configuration defining a node.

The second step consists in smoothing these stop motions to obtain a shorter trajectory in time TR. Smoothing uses the same 3D model than the research phase of the path. Thus, collisions are tested during the computation of the transition moves at each node. If a collision appears during the smoothing of the stop move at node q_i, then the movement will not be smoothed for this node and the stopping move will be kept.

In the following, we detail a method for smoothing stop motions based on the computation of a fixed time movement using the *3-segment* method presented in previous work.

4.4.1 Smoothing of the Stop Motions

We propose a method based on the minimum time algorithm for trajectory generation (Sect. 4.3.2) [12] and on the *3-segment* method (Setion 4.3.3) to smooth the stopping motions [11].

The trajectory TR_{ptp} (Fig. 17) between the first two nodes q_{init} and q_1 is a point-to-point motion in a straight line of duration $T_{(q_{init}q_1)}$. Similarly the motion between q_1 and q_2 is a point-to-point motion of duration $T_{(q_1q_2)}$. The stop motion is smoothed between the points $M_{1,2}$ et $M_{2,1}$.

Notation: We note the points that limit the smoothing $M_{i,j}$, the index i is the index of the point-to-point motion (the first of the trajectory has an index of 1). The index $j \in \{1,2\}$ is 1 if this point is the final extremity of the transition motion with the previous point-to-point motion and conversely for $j = 2$.

Choice of the Points $M_{i,j}$

Let us consider the transition motion in the neighborhood of q_1 located at time t_{q1}:

$$t_{q1} = t_I + T_{(q_{init}q_1)} \tag{11}$$

To simplicity, we choose $t_I = 0$ as the time origin of the trajectory.

The time positions $t_{M_{1,2}}$ and $t_{M_{2,1}}$ of the points $M_{1,2}$ and $M_{2,1}$ are determined from a given parameter τ such that:

$$M_{1,2} = TR_{ptp}(t_{q1} - max(\tau, \frac{T_{(q_{init}q_1)}}{2})) \tag{12}$$

$$M_{2,1} = TR_{ptp}(t_{q1} + max(\tau, \frac{T_{(q_1q_2)}}{2})). \tag{13}$$

So when τ is null, the movement stops at the point q_1. When τ satisfies (14), the transition motion connects the midpoints of the line segments (q_{init}, q_1) and (q_1, q_2) because of the symmetry of the velocity profile about this point.

$$\tau \geq max\left(\frac{T_{(q_{init}q_1)}}{2}, \frac{T_{(q_1q_2)}}{2}\right) \tag{14}$$

In practice, unless otherwise specified, by default we choose the points $M_{i,j}$ such that the transition movement begins at the end of the constant velocity segment of the first point-to-point movement (P_1, P_2); the transition movement ends at the beginning of the constant velocity segment of the second point-to-point movement (P_2, P_3).

Notice that, for a given value of the parameter τ, the Euclidean distance between the points $M_{i,j}$ and the corresponding point q_i varies according to kinematic parameters of the point-to-point movement.

4.4.2 Computation of the Transition Movement

Let us consider a trajectory of dimension n. The instants $t_{M_{i-1,2}}$ and $t_{M_{i,1}}$, start and end of the transition movement at the configuration q_i, are identical for all n dimensions. The computation method is described by Algorithm 1. The first step consists in computing, for each axis, the optimal time motion to determine the duration T_{imp} of the transition movement. The method *3-segment* to compute the movement in fixed time is then applied to each axis.

Algorithm 1: Computation of a transition movement near of a node q_i

begin
  Determining the switching points $M_{i-1,2}$ et $M_{i,1}$ (eq. 13 and 12)
  for *each dimension* n_i **do**
    Computation of the one-dimensional movement in minimum time (Section 4.3.2)
    Computation of the duration of the one-dimensional movement in minimum time $T_{opt}[i]$
  end
  Determination of the duration of the transition movement
  $T_{imp} = max(\forall\, i \in [1,n] \mid T_{opt}[i])$
  for *each dimension* n_i **do**
    Computation of triplets of cubic curve segments from the method *3-segments* (Section 4.3.3)
  end
end

Figure 18 illustrates an application of the method for the case of a movement defined by three points P_1, P_2, P_3 and by the kinematic constraints $V_{max} = 0.1m/s$, $A_{max} = 0.3m/s$ et $J_{max} = 0.9m/s$. The transition movements are computed for different values of the parameter τ.

The proposed method ensures the continuity in velocity and acceleration for each dimension. The initial and final velocities of the transition movements can be different and acceleration not zero. The duration of the transition movement is computed by taking into account the kinematic constraints of each dimension using the minimum time algorithm (Sect. 4.3.2). Therefore this method guarantees that changes in velocity, acceleration and jerk are limited. However, in some cases, constraints can be exceeded by the *3-segment* method. In practice, we introduce a percentage (10%) of exceeding for each constraint. If, for a transition movement, the exceeding of kinematic constraints

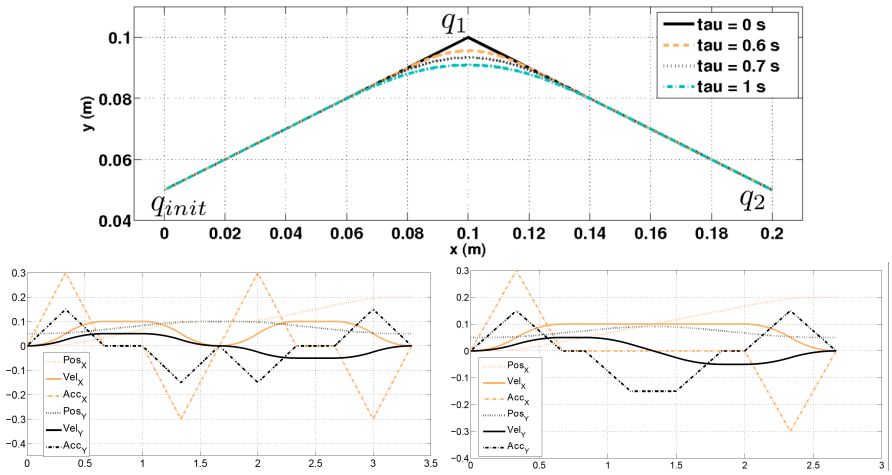


Figure 18: Transition movement for two lines that form an angle of about 127 (top), graph of position, velocity and acceleration as function of time for a point-to-point motion for $\tau = 0$ (bottom-left) and $\tau = 1s$ (bottom-right)

is too large, this movement is not smoothed to comply with the constraints of human comfort.

4.5 Application to Robot Manipulators

To better explain the method, we apply it to an example of task of grasping an object, the grey tape cassette of the Fig. 19. The path of the center of the end effector of the robot (hand) is described by the green line segments in Fig. 20. On this path, the spheres represent the initial, final and intermediate configurations (nodes). The path of the point-to-point trajectory TR_{ptp} is identical to the path planned. This trajectory stops at each intermediate node. The smoothed path TR is represented by the black curve. We note that the trajectory stops at the first node as a smoothing in its neighborhood would have introduced to a collision[2] between the hand of the robot and the environment. The planning of the path of the trajectory was performed in the Cartesian space of the robot by considering the platform was fixed. The following section presents the methodology to take into account the redundancy of the robot.

4.6 Planning in the Cartesian Space

4.6.1 Generation of the smoothed trajectory TR in Cartesian space

To represent the complete configuration of the robot in Cartesian space, we propose to use a vector X_i with:

[2]Note: Another solution would be to compute a path that goes farer from the obstacle but it is not the purpose here.

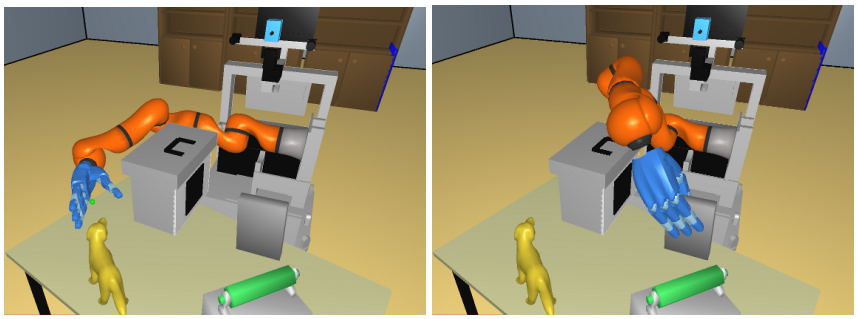


Figure 19: Initial configuration and grasp configuration of the robot Jido.

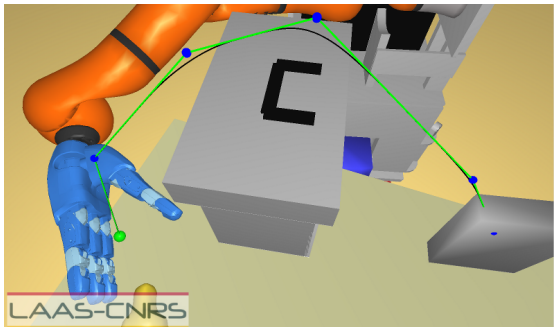


Figure 20: Trajectories TR_{ptp} et TR in the Cartesian space to grasp the cassette

- the position of the robot base,
- the pose of the end effector(s),
- the configuration of the redundancy axis of the arms if they have more than six degrees of freedom (DOFs),
- the configuration of the hand(s),
- the configuration of the head.

In the following, we consider that the platform is fixed. For a system operating in 3D space, six independent parameters are used to define the position of the end effector. For the planning, the system is decomposed into *passives* and *actives* parts corresponding respectively to dependent and independent variables [24, 35]. Thus a robot manipulator with six DOFs, is decomposed as follows: the independent variables (active) are the six DOFs (position and orientation) of the end effector and the joint variables are the dependent variables (passive) .

In the case of our Jido[3] robot, as the robot arm is composed of seven DOFs, it is therefore redundant. In addition to the pose of the end effector, an articulation of the

[3]Jido is an MP-L655 platform from Neobotix, equipped with a KUKA LWR arm.

arm is chosen and becomes an *active* variable. Notice that, if the motion of a holonomic platform was considered, then these DOFs would be *active* variables.

During the planning of the path in the Cartesian space, only the *active* variables are sampled using, according to the circumstances, the RRT or the T-RRT algorithm. The *passive* variables are computed in a second step by solving the inverse kinematics of the arm prior to test the validity of the sampled configuration of the robot (bounds and collision) (See section 4.1.1). During the test of the validity of a local path between two configurations, the inverse kinematic function is also called.

To perform the interpolation between two configurations, we represent the position of the end effector by a displacement: three parameters for the position and three parameters for the orientation (vector and angle representation with the norm of the vector equal to the angle [12]). We have implemented a local method of interpolation between two configurations. This method takes as parameters two local configurations (with their kinematic conditions) and the imposed kinematic constraints (J_{max}, A_{max} et V_{max}) for each active axis. After applying the local method between each intermediate configuration, the obtained trajectory TR_{ptp} is composed of point-to-point movement of dimension n (n is the number of active axes), that is for Jido $n = 22$ parameters (6 for the end effector, 1 for the axis of the redundant manipulator, 13 for hands and 2 for the head).

The smoothed trajectory TR in Cartesian space is then obtained by the method described in the previous section applied to the active axes (Sect. 4.4).

4.6.2 Conversion of the Trajectory in the Joint Space of the Robot

As most of the robot controllers operate in the joint space, it is important to provide a solution to convert Cartesian trajectories into joint ones. To perform this transformation, the trajectories of passive axes are obtained by discretizing the trajectory TR defined in Cartesian space and performing inverse kinematics for each sample. The trajectory TR is discretized at the period of operation of the robot controller. This allows obtaining the position, and by derivation, the velocity and the acceleration of all the DOFs of the robot.

However, this discretization removes the notion of time and requires a large amount of data to represent the trajectory.

We can use the approximation method of trajectory presented in Section [11] and [10] to approximate this discretized trajectory and thus obtain a compact description of the trajectory. Unlike the approximation in the Cartesian space, the trajectory error taken into account by the approximation algorithm is the maximum error of the trajectory of each DOF.

The obtained approximated trajectory TR_{app} is a function of time, it is composed of series of segments of cubic curves for each joint variable of the robot.

However, movements of the passive axes are not planned, they can exceed the kinematic limits of the robot. In this case, the trajectory cannot be directly performed. To adapt the trajectory when the task allows it, we replace the time parameter t of the trajectory by applying a function α, $\mathbb{R} \longrightarrow \mathbb{R}$. The function α will make it possible to change the time increment during the execution of the trajectory and therefore allow slowing down the execution.

The period of the trajectory controller is denoted ΔT. In the case of a classical execution, the application α is defined by:

$$\alpha(t) = t \tag{15}$$

or, in discrete notation:

$$\alpha(k\Delta T) = \alpha((k-1)\Delta T) + \Delta T \tag{16}$$

The trajectory carried out is $TR_{app}(\alpha(t))$.

The introduction of the function α makes it possible to modify the motion law of the trajectory TR_{app} and thus to adapt the evolution of each joint of the robot in a synchronized way.

To determine the function α in the case where one wishes to adapt the motion law, we first determine for each instant of the trajectory TR_{app} exceeding β the velocity of each axis relatively to the corresponding maximum velocity (maximum values used here are the default limits accepted by the system). We obtain:

$\forall k\Delta T \in [t_I, t_F]$,

$$\beta(k\Delta T) = \begin{cases} 1 & if\ \forall j \in [1,n],\ {}_jV(k\Delta T) \leq_j V_{max}^{mot} \\ min\left(\forall j \in [1,n] \mid \frac{{}_jV_{max}^{mot}}{{}_jV(k\Delta T)}\right) & \text{sinon} \end{cases} \tag{17}$$

where n is the number of controlled DOFs, ${}_jV(t)$ the evolution of the velocity of the articulation j and ${}_jV_{max}^{mot}$, the maximum velocity of the articulation j.

Thus we obtain:

$$\alpha(k\Delta T) = \alpha((k-1)\Delta T) + \beta(k\Delta T)\Delta T \tag{18}$$

with $\alpha(0) = 0$.

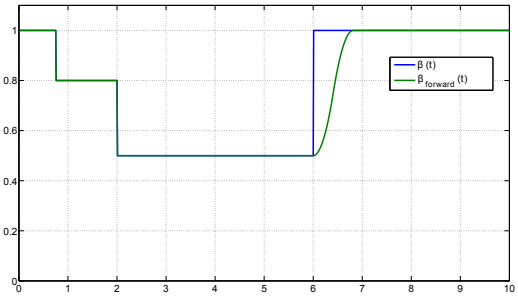
However, the trajectory $TR_{app}(\alpha(t))$ cannot be executed directly because it would introduce discontinuities in velocity due to the discontinuity of β. To smooth the evolution of β, we apply a variant of the method described in Sect. 4.3.4 that anticipates the change in β. The smoothed function β is denoted by β_{smooth}. The smoothing is performed in three steps by the Algorithm 2.

Algorithm 2: Smoothing the variation of the motion law

begin

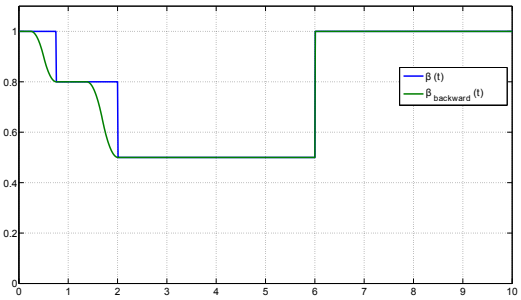
Applying the method in Sect. 4.3.4 on the evolution of β by varying the time from t_I to t_F with a step of ΔT, the resulting curve is named $\beta_{forward}$:

$$\beta_{forward}(k\Delta T) = \begin{cases} \beta(k\Delta T) & if\ \beta(k\Delta T) < \beta((k-1)\Delta T) \\ f_{smooth}(\beta) & \text{otherwise} \end{cases} \tag{19}$$



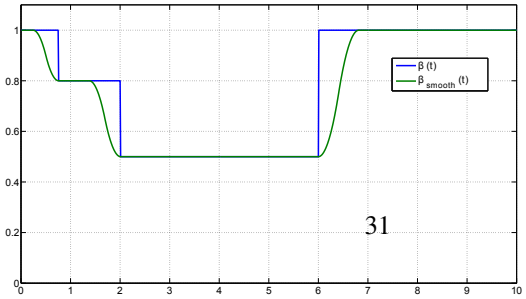
Applying the method in Sect. 4.3.4 on the evolution of β by varying the time from t_F to t_I with a step of $-\Delta T$, the resulting curve is named $\beta_{backward}$:

$$\beta_{backward}(k\Delta T) = \begin{cases} \beta(k\Delta T) & if\ \beta(k\Delta T) > \beta((k+1)\Delta T) \\ f_{smooth}(\beta) & \text{otherwise} \end{cases} \tag{20}$$



β_{smooth} is finally obtained by taking the minimum between $\beta_{forward}$ and $\beta_{backward}$:

$$\beta_{smooth}(k\Delta T) = \min(\beta_{forward}(k\Delta T), \beta_{backward}(k\Delta T)) \tag{21}$$



end

The method presented above allows modifying the velocity of each joint of the robot to satisfy the velocity bounds. We have supposed that the resulting path respects the constraints of acceleration. Otherwise, it is possible to identify a function β^{acc} equivalent to β to take into account overtaking accelerations. In practice, for HRI, the kinematic constraints of the trajectory are small in comparison to the capabilities of the system and it is not necessary to check for overtaking of acceleration.

In this section we have presented a method to compute a trajectory from a path that can be defined either in the joint space or in Cartesian space of the robot.

4.7 Adaptation of the Motion Law During the Execution

During the trajectory execution, humans located in the workspace of the robot can move, so the robot can put them in danger. We propose to use the geometric models of the robot and of the human, updated at each iteration during the execution to ensure the safety and comfort of humans. We choose to take into account the weighted average *cost* of the security and visibility constraints introduced in Sect. 4.1.2. The method to adapt the motion law is the same as the one presented in the previous section. The costs are high when the distance human-robot is short or when the robot is outside the field of view of the human, the cost taken into account is $cost_{inv} \in [0,1]$ such that:

$$cost_{inv}(k\Delta T) = 1 - cost(k\Delta T) \tag{22}$$

The cost $cost_{inv}$ is then smoothed on-line by the function f_{smooth} presented in Sect. 4.3.4.

When the trajectory $TR(\alpha(t))$ is planned in Cartesian space, the law $\alpha(t)$ is evaluated at each iteration:

$$\alpha(k\Delta T) = \alpha((k-1)\Delta T) + f_{smooth}(\min(cost_{inv}(k\Delta T), \beta_{smooth}(k\Delta T)).\Delta T \tag{23}$$

In this section, we have presented a human aware motion planner. In a first part we have introduced some elements to take into account the relative position of the robot and the human, and the human behavior. Using a ramdom motion planner and cost map to represent the human constraints, this motion planner begins by computing a broken line path that is then transformed in feasible trajectories. The trajectory generator allows limiting velocity, acceleration and jerk. This generator is integrated in the motion planner and firstly presented in the case of planning in the configuration space. It is then extended to planning in Cartesian space.

The approach is general and can be applied to complex systems with two hands/arms. We have proposed an original method to convert a Cartesian trajectory in a joint trajectory. Finally, we have presented an approach to modify online the evolution of the time law of the trajectory and shown its usefulness for taking into account the presence of humans during the execution of the movement.

In the next section, we introduce an attentional system to monitor the robot activity from the perspective of the software components.

5 Attentional System

A robotic system designed to physically interact with humans should adapt its behavior to the human actions and the environmental changes in order to provide a safe,

natural, and effective cooperation. The human motions and the external environment should be continuously monitored by the robotic system searching for interaction opportunities while avoiding dangerous and unsafe situations. In this context, attentional mechanisms [53, 21] can play a crucial role: they can direct sensors towards the most salient sources of information, filter the available sensory input, and provide implicit sensory-motor coordination mechanisms [41] to orchestrate and prioritize concurrent activities.

In this project, we have deployed an attentional system suitable for balancing the trade off between safe human-robot interaction and effective task execution. The attentional system is to supervise and orchestrate the human-robot interaction activities monitoring their safety and effectiveness. Our attentional execution monitoring system is obtained as a reactive, behavior-based system, endowed with simple, bottom-up, attentional mechanisms. We assume a frequency-based model of the executive attention [15, 16, 13] where each behavior is endowed with an adaptive internal clock that regulates the sensing rate and action activations. The frequency of sensor readings is here interpreted as a degree of attention towards a behavior: the higher the clock frequency, the higher the resolution at which the behavior is monitored and controlled. In particular, we consider robot manipulation tasks providing the attentional monitoring strategies for behaviors like pick and place, give and receive, search and track (humans and salient objects).

5.1 Related Work

Human aware manipulation [59, 61] and human-robot cooperation in manipulation tasks [27] are very relevant topics in HRI literature, however cognitive control and attentional mechanisms suitable for safe and effective interactive manipulation are less explored. A number of recent contributions about close HRI are based on motivational and cognitive models [8]. However, attentional mechanisms in HRI have been mainly investigated focusing on visual and joint attention [50, 8] for social interaction. In contrast, our main concern is on (supervisory) executive attention for monitoring and action orchestration [53, 21]. Attentional mechanisms applied to autonomous robotic systems have been proposed in the literature for vision-based mobile robotics (e.g. [49, 19, 31]), but here we are interested in artificial attentional processes suitable for monitoring the execution of multiple concurrent behaviors in human-interaction tasks [14].

5.2 Attentional Model

Our aim is to develop an autonomous robotic system suitable for human-robot interaction in cooperative manipulation tasks. Achieving autonomy and safety in such an environment requires adaptation. For this purpose, we propose to deploy an attentional system, a kind of supervisory attentional system a la [53], to modulate the robotic arm motion and perception. The attentional system is expected to monitor and regulate multiple concurrent activities [41] in order to achieve an effective coordination and interaction with the human movements in the operative space. More specifically, our attentional model combines the following design principles:

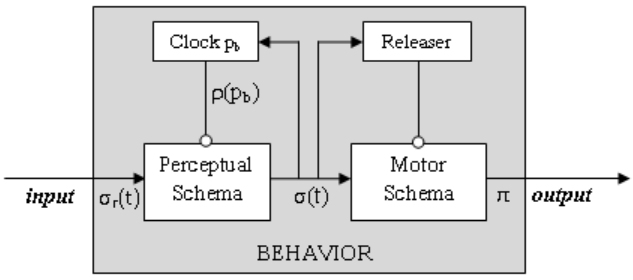


Figure 21: AIRM Model: each behavior is composed of an adaptive clock, a releasing function, a perceptual schema and a motor schema.

- *Behavior-based system*. The executive control is obtained from the interaction of a set of multiple parallel behaviors working at different levels of abstraction.
- *Attentional monitoring*. Attentional mechanisms are to focus monitoring and control activities on relevant internal behaviors and external stimuli.
- *Internal and external sources of salience*. The sources of salience are behavior and task dependent; these can be dependent by either internal states (e.g. resources, processes, goals) or external stimuli (e.g. obstacles, unexpected variations of the environment).
- *Adaptive sensory readings*. For each behavior, the process of changing the rate of sensory readings is interpreted as an increase or decrease of attention towards a particular aspect of the environment the robotic system is interacting with.
- *Emergent attentional behavior*. The overall executive attention should emerge from the interrelations of the attentional mechanisms associated with the behaviors.

5.2.1 A Frequency-based Model of Attention

The frequency-based model of the executive attention [15, 16] adopted in this chapter can be represented in a schema theory framework in terms of Adaptive Innate Releasing Mechanisms (AIRMs) [16]. In the following we briefly recall this model.

In Fig. 5.2.1, the AIRM is represented through a Schema Theory representation [2], where each behavior is composed of a Perceptual Schema (PS), which reads and processes incoming data from sensors, a Motor Schema (MS), producing commands to be given to motors, and a control mechanism, based on a combination of a *releasing mechanism* [64] and an internal *adaptive clock*. In particular, the releaser acts as a trigger signal that enables or disables the activation of the MS, according to the sensory data $\sigma(t)$. For example, a detected obstacle releases the obstacle avoidance MS. Instead, sensor readings are sampled by the adaptive clock. That is, the robot reads data just when necessary (reducing sensory readings and elaborations) with a period

that can change according to the salience of the perceived stimuli. In this way, each behavior can independently monitor the environment and modulate its outputs following the clock frequency changes.

Assuming a discrete time model for the adaptive clock, the way the clock adapts its period is called *monitoring strategy* and it is characterized by:

- A period p^t for each behavior, ranging in an interval $[p_{min}, p_{max}]$,
- An *updating function* $f(t) : \mathbb{R}^n \rightarrow \mathbb{R}$ that changes the clock period p^t, according to the parameters the behavior depends on (sensors, internal state, environmental features, and the behavioral goal).
- A trigger function $\rho(t, p^{t-1})$, which enables/disables the data flow $\sigma_r(t)$ from sensors to PS, at every p^{t-1} time unit:

$$\rho(t, p^{t-1}) = \begin{cases} 1, & \text{if } t \bmod p^{t-1} = 0 \\ 0, & \text{otherwise} \end{cases} \tag{24}$$

- Finally, a support function $\phi(f(t)) : \mathbb{R} \rightarrow \mathbb{N}$ that maps the values generated by the updating function $f(t)$ in the allowed range for the period $[p_{min}, p_{max}]$:

$$\phi(x) = \begin{cases} p_{max}, & \text{if } x \geq p_{max} \\ \lfloor x \rfloor, & \text{if } p_{min} < x < p_{max} \\ p_{min}, & \text{if } x \leq p_{min} \end{cases} \tag{25}$$

Now, starting from the clock period at time 0, $p^0 = p_{max}$, the clock period at time t is regulated as follows:

$$p^t = \rho(t, p^{t-1}) * \phi(f(t)) + (1 - \rho(t, p^{t-1})) * p^{t-1}. \tag{26}$$

That is, if the behavior is disabled, the value of the clock period at time t remains unchanged at the previous value p^{t-1}. Instead, when the trigger function is equal to 1, the behavior is activated and, subsequently, its activation period changes according to the $\phi(f(t))$ function.

5.2.2 Attentional HRI

Based on the model introduced above, we have designed a behavior-based control system endowed with attentional monitoring strategies for human-robot interaction. In this model, the attentional mechanisms regulates the executive system trading off between two conflicting requirements:

- safe interaction with the humans;
- effective cooperation in interactive tasks.

Each requirement is associated with a motivational drive that affects the attentional and executive state of the robotic behavior. The first one corresponds to the fear of hurting people, hence it determines caution, slow movements and intensive monitoring

(in case of danger it blocks the robot motion), instead, the second one is associated with a desire to interact with people and manipulate objects, thus this attitude provides an attraction towards moving and close persons or objects.

Depending on the disposition, movements, and the attitude of a person in the robot workspace, each behavior changes its activation frequency, affecting the overall attentional state of the system. In this way, a person walking across the interaction area or a fast movement of a human head (or hand) can modify the behaviors' attentional state causing an accelerated elaboration of the associated perceptual input (human movements) and more frequent behaviors' activations.

Test-bed domain. We have considered a robotic manipulator that is to cooperate with humans in pick-and-place and give-and-receive (hand-over) tasks. Depending on the context, the robotic system should: look for an operator to interact with; give or receive an object to/from the operator; pick or place an object from/into a location. Each of these tasks are to be monitored in order to avoid dangerous/unsafe situations.

In this context, the attentional mechanisms allow us to combine the robot attraction towards human operators (to be effective and cooperative) and the robot repulsion from unexpected events and abrupt environmental changes. For each behavior, the simple perception-action response to an external stimulus may produce different patterns of interactions depending on different internal states of the robot given by the combination of the fear of hurting the user and the desire of helping him.

Environment. In our setting, the robot base is kept fixed (the mobile base is not exploited) and close to a small table where the robot can pick and place objects. Depending on the proximity, we have defined three areas in the workspace: a proximity area which is too close to the robot body and unsafe for HRI; an interaction area, where physical human-robot interaction is possible (here we refer to both visual and physical interaction in the robotic arm workspace); a far workspace area where humans and object are in the robot field of view, but too far for handover tasks.

5.3 Control Architecture

We have designed a control architecture suitable for the primitive interactive manipulation tasks introduced above. The control system integrates modules for forward, inverse kinematics, and visual servoing along with modules for face recognition, hand detection/tracking, object recognition/tracking. Given these functionalities, the attentional state of the robot is affected by the following sources of saliency: face, hands, object detection, proximity.

5.3.1 Attentional Behaviors

The behavior-based architecture is depicted in Fig. 22. This model integrates attentional behaviors for pick and place, give and receive, but also behaviors for search and track (humans and objects) as well as behaviors regulating the avoid attitude of the robotic system.

The robot attentional behavior is obtained as the combination of the following primitive behaviors (see Fig. 22): `AVOID`, `PICK` and `PLACE`, `GIVE` and `RECEIVE`,

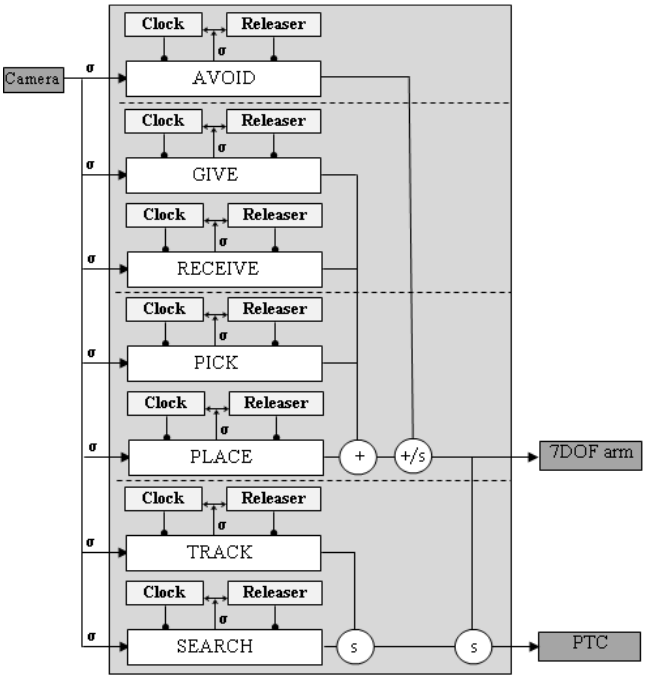


Figure 22: Behavior-based architecture for HRI.

SEARCH and TRACK. For each behavior, we have to define the activation function and the updating policy that represents the associated attentional model.

Behaviors settings. SEARCH controls the pan-tilt (PTU) providing an attentive scan of the environment looking for humans and objects. It is active whenever the robotic system is idling and no interesting things (objects or humans) are in the robot field of view. Its activation is periodic, but not adaptive, hence it is associated with a constant clock:

$$p_{sr}^{t} = k_{sr}. \tag{27}$$

Once a human is detected in the robot workspace (through face detection and/or hand detection), the TRACK behavior is enabled. This behavior allows the robot to monitor human motions before they enter in the interaction space. TRACK focuses the system attention on the operator movements, hence the adaptive clock should be regulated in accordance with the human motion and position. Here, the input signal $\sigma_{hm}(t)$ represents the human distance from the robot camera, in our test-bed it is the minimal distance of human faces and hands. The TRACK clock period changes according to $\sigma_{hm}(t)$ and the increment of $\sigma_{hm}(t)$, that is, the period p_{tr} is updated as follows:

$$p_{tr}^{t} = \Theta_{tr}(\sigma_{hm}(t), \frac{\sigma_{hm}(t) - \sigma_{hm}(t - p_{tr}^{t-1})}{p_{tr}^{t-1}}), \tag{28}$$

where p_{tr}^{t-1} is the period at the previous clock cycle, $\Theta_{tr}(x,y)$ is a a function $\Theta_{tr}(x,y) = \phi_{tr}(\alpha x + (1-\alpha)1/y + \beta)$, where α and β are behavior-specific parameters used to weight the importance of position and velocity in the attentional model, while $\phi_{tr}(z)$ is the scaling function that introduces suitable thresholds to keep the clock period within the allowed interval $[p_{tr\_min},\ p_{tr\_max}]$. Intuitively, a human that moves fast and close needs to be carefully monitored (high frequency, foreground), while a human that moves far and slow can be monitored in a more relaxed manner (low frequency, background).

The `AVOID` behavior checks for safety in human-robot interaction, it controls the arm motion speed and can stop the motion whenever a situation is assessed as dangerous. `AVOID` is enabled when a human is detected in the robot interaction area. It is endowed with an internal clock whose frequency depends on the operator proximity and motion. The associated clock frequency changes proportionally to the situation saliency. That is, if the operator is close and/or its position σ_{op} (i.e. minimal distance of face and hands) becomes closer between successive readings of sensory data, then the clock is accelerated, while it is decelerated if the operator moves away from the robot. The period of this clock changes as follows:

$$p_{av}^{t} = \Theta_{av}(\sigma_{op}, \frac{\sigma_{op}(t) - \sigma_{op}(t - p_{av}^{t-1})}{p_{av}^{t-1}}), \tag{29}$$

where Θ_{av} is defined as for `TRACK`. The output of this behavior results in a speed deceleration associated with high frequencies:

$$speed = \begin{cases} \frac{max\_speed \times p_{av}^{t}}{p_{av\_max}} & \text{if } prox.sp. < \sigma_{op} \leq inter.sp. \\ 0 & \sigma_{op} \leq prox.space \end{cases} \tag{30}$$

where $speed$ is the current speed, max_speed is the maximum allowed value for the arm speed, $prox.sp.$ and $inter.sp.$ are the proximity and the interaction space respectively. Moreover, the arm will stop if the operator is inside the robot proximity space.

The `PICK` behavior is activated when the robot is not holding an object, but there exists a reachable object in the robot interactive space. `PICK` moves the robot's end-effector towards the object, activates a grasping procedure and, once the robot holds the object, moves this in a predefined safe position close to the robot body. For `PICK`, the input signal $\sigma_{obj}(t)$ represents the distance of the object from the robot end effector which can be detected by the stereo camera. In this case the clock period is associated with the distance of the object. That is, the period p_{pk}^{t} is updated as follows:

$$p_{pk}^{t} = \phi_{pk}(\alpha\, \sigma_{obj}(t)), \tag{31}$$

with $\phi_{pk}(x)$ is the scaling function used to scale and map $\sigma_{obj}(t)$ in the allowed range of periods $[p_{pk\_min},\ p_{pk\_max}]$. Furthermore, the clock frequency determines also speed variations. In particular, the speed is related to the period according to the following relation:

$$speed = \frac{max\_speed \times p_{pk}^{t}}{p_{pk\_max}}, \tag{32}$$

In this way, the arm moves with *max_speed* at the beginning, when there is free space for movements (and a low monitoring frequency), and smoothly reduces its speed to a minimum value in order to execute a precision grip with more frequent camera information (higher monitoring frequency).

As for PLACE, it is activated when the robot is holding an object in the absence of interacting humans in the interactive space. It moves the robot end effector towards a target position, it places the object and moves the robot arm back to a predefined position close to the robot body. The clock period is regulated by a function analogous to that of (31) with the distance to the target σ_{tr} as the input signal. Also in this case, the speed is decelerated at high clock frequencies according to (32).

The GIVE and RECEIVE behaviors are activated by object and gesture detection. These behaviors are responsible for monitoring and regulating the activities of giving and receiving objects taking into account both the humans' proximity and their movements. In this case, the clock period is associated with the distance of both the objects and the speed of the operator hand. In particular, GIVE is activated when the robot holds an object and perceives a reachable human hand in its operative space. When activated, this behavior moves the end effector in the direction of the operator's hand with a trajectory and velocity which depends on the human's proximity and operator's hand movements. The GIVE sampling rate is regulated by the following function:

$$p^t_{gv} = \Theta_{gv}(\ \ \gamma_{obj}(\|\sigma_{obj}(t) - ee_{pos}(t)\|), \gamma_{op}(\frac{\sigma_{op}(t) - \sigma_{op}(t - p^{t-1}_{gv})}{p^{t-1}_{gv}})\ \ ), \tag{33}$$

where $\sigma_{obj}(t)$ and $ee_{pos}(t)$ are the positions of the object and the end effector at time t, $\sigma_{op}(t)$ is the hand operator position, θ_{gv}, γ_{obj}, γ_{op} are suitable functions defined as follows. The function γ_{obj} sets the period proportional to the object position, i.e., the closer the object, the higher the sampling frequency:

$$\gamma_{obj} = (p_{gv\_max} - p_{gv\_min})\frac{d}{max_d} + p_{gv\_min}, \tag{34}$$

with d, max_d are, respectively, the distance $(\sigma_{obj}(t) - ee_{pos}(t))$ and the maximal distance between the end effector and the object. Instead, γ_{op} depends on the hand speed v (in terms of the incremental ratio of the hand position towards the value of the period), i.e., the higher the speed, the higher the sampling frequency. The following function is used to set and normalize the values within the allowed interval $[p_{min}, p_{max}]$:

$$\gamma_{op} = \begin{cases} (p_{gv\_max} - p_{gv\_min})(1 - v) + p_{gv\_min} & \text{if } v \leq 1 \\ p_{gv\_min} & \text{otherwise} \end{cases} \tag{35}$$

Finally, the $\Theta_{gv}(x)$ combines the two functions γ with a weighted sum regulated by an α parameter

$$\Theta_{gv}(x) = \phi_{gv}(\alpha\gamma_{obj} + (1 - \alpha)\gamma_{op})), \tag{36}$$

also in this case the resulting period is limited within the allowed interval $[p_{gv\_min}$, $p_{gv\_max}]$ by the scaling function ϕ_{gv}.

The clock frequency regulates not only the sampling rate, but also the velocity of the arm movements. More specifically, the execution speed is related to the period

according to an inversely proportional relation (32). This means that the higher the sampling rate, hence the attention, the slower the hand movement. Intuitively, here we assume that when attention is needed the movement should be more carefully monitored, thus slower.

As for the RECEIVE behavior, it is activated when the robot perceives a human in the operative space holding a reachable an object in his/her hand. The behavior sampling rate is regulated by a function analogous to (33) (set with different parameters) with an adaptive velocity inversely proportional to the current period, as in (32).

5.4 Execution Example

We now illustrate how the system works in typical interactive situations. In Fig. 23 we plotted part of the execution of the RECEIVE behavior. In particular, Figure 23.a represents the variation of the distance between the end effector of the robotic arm and the operator's hand. In the execution cycle 80, the robot has almost reached the human hand, however the operator moves his/her arm away. The execution of the behavior ends at the execution cycle 162 when the robot delivers the object to the operator. Figure 23.b represents the hand speed variation of the same execution, as evaluated by the RECEIVE behavior. The hand is almost stationary between the cycle 30 and the

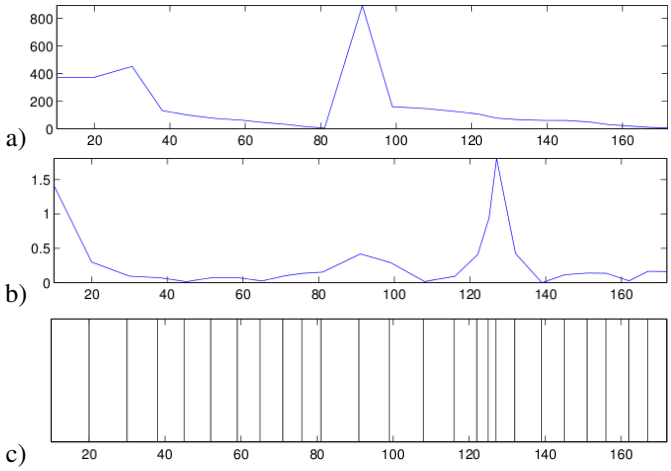


Figure 23: a) End effector-hand distance; b) Hand speed as evaluated by the Receive Behavior; c) Activations of the Receive behavior.

cycle 70, then it starts moving with different speeds until it stands still at cycle 162 and receives the object. Finally, Figure 23.c represents the activations of the behavior at each cycle. Whenever there is a bar in the plot, this means that the behavior perceptual schema is active. Let us note that both the distance and the hand speed are sampled and evaluated only when the behavior perceptual schema is active. The frequency of activation will increase when the distance is small (for example between cycles 40 and

80) or when the hand speed is high (for example between cycles 105 and 125) following the updating function of the behavior.

5.5 Evaluation Criteria and Experimental Results

To evaluate the performance of the attentional system and of the HRI system, we introduce some evaluation criteria considering safety, reliability, effectiveness and efficiency.

- *Safety* is measured counting dangerous human-robot interaction events (i.e. a safe robot should avoid collisions between human and a moving robot and it should minimize interactions where the two are too close).
- *Reliability* is evaluated considering unrecoverable world/robot states encountered during the tests (the robot is stuck, the object falls down, the object is not reached or located by the robot).
- *Effectiveness* is assessed considering the time needed to achieve the task (the system should minimize the time to achieve the task).
- *Efficiency* is associated with the number of behavior activations needed to achieve the task (for us, an attentional system is efficient, when it can distribute computational resources among different processes, focusing only on relevant activities).

Parameters Setting. Given the attentional model introduced in the previous section, the overall attentional behavior is obtained once we tune the parameters associated with the behaviors' monitoring strategies.

To assess the performance of the system with respect to the previous set of criteria we introduc a suitable optimization function:

$$f = \quad M1 \times N_{Safe} + M2 \times N_{Rel} + M3 \times T_{Effe} + M4 \times N_{Effi}.$$

Here, $M_1 > \cdots > M_4$ specify the priorities in terms of weights; N_{Rel} represents the number of unrecoverable situations with respect to the number of accomplished activities (pick, place, etc.); N_{safe} the HRI unsafe situations with respect to the executed activities; T_{Effe} is for the time spent to achieve the tasks with respect to the overall mission time; N_{Effi} is for the number of behavior activations with respect to the maximal possible activations (for each behavior p_{min}).

This function can be exploited, on the one hand, to learn the system parameters and, on the other hand, to validate the overall system behavior. Different learning algorithms can be deployed for parameter learning (e.g. genetic algorithms, particle swarm optimization, simulated annealing etc.), currently, we are investigating Differential Evolution algorithms (DE) [62] which are particularly suitable for both boundedness and granularity problems, indeed DE manages unrestricted and unbounded range of values. More details about DE methods used to set attentional monitoring strategies can be found in [13].

	Effectiveness		Efficiency		Reliability		Safety	
	AIRM	P1VMED	AIRM	P1VMED	AIRM	P1VMED	AIRM	P1VMED
Receive	$7.66s \pm 0.54s$	$9.69s \pm 0.31s$	14.5 ± 1.57	41.7 ± 1.42	100%	100%	100%	100%
Give	$4.87s \pm 1.4s$	$7.27s \pm 2.9s$	6.05 ± 2.65	14.65 ± 5.59	83%	80%	90%	84%
Pick	$9.14s \pm 2.07s$	$10.48s \pm 0.67s$	16.2 ± 6.58	32.65 ± 5.99	77%	54%	100%	100%
Place	$6.03s \pm 1.05s$	$8.96s \pm 0.6s$	12.95 ± 5.03	58.65 ± 10.17	100%	100%	100%	100%

Table 1: Evaluation of the Effectiveness, Efficiency, Reliability, and Safety criteria.

Experimental Setup. In order to evaluate the performance of the AIRM architecture we compare it with a classical non-rhythmic architecture (P1Vmed) in which the behaviors perceptual schema are always active. For the adaptive version (AIRM) we consider adaptive concurrent clocks with $p_{min} = 1$, $p_{max} = 10$ and $speed = \frac{max - speed \times p}{p_{max}}$ for all the behaviors. For the (P1Vmed), we assume that the behaviors' perceptual schema are always active (i.e., p_{min} and p_{max} are both equal to 1) and the arm speed is set to a constant value ($speed = \frac{max - speed}{2}$). Moreover, in the case of the AIRM architecture, the updating policies of the behaviors are those specified in the previous section. The range of values for the speed is $[0; 0.3]$ m/s. For the experiments, we have used the UNINA robotic platform, endowed with a 7DOF robotic arm (Cyton Arm by Energid: payload 300 g, hight 60 cm, reach 48 cm, joint speed 60 rpm), a gripper (size 3.25 cm) as end effector, and a kinetic device. In this context, the proximity, interaction, and workspace distances were set, respectively, at 20 cm from the robot body, $10 - 50$ cm, and 50 cm to 6 m.

Empirical Results. During the empirical evaluation, we have tested each behavior 20 times with 5 different operators unaware of the robot behavior. Operators are required to observe the robot and move around in the case of Pick and Place behaviors, and interact, without any specific requirement, for the Give and Receive behaviors. In these final cases all the hand movements, made by the operators, were spontaneous. For each test we have evaluated the parameters defined above: effectiveness, efficiency, reliability, and safety.

Notice that not only are the attentional mechanisms associated with better performance in terms of effectiveness and efficiency (Fig. 24 and Table 1), but we also observed better results regarding reliability and safety (Tab. 1) compared with the non-adaptive architecture in which the perceptual schemas are always active (P1Vmed). In particular, the adaptive modulation of the robotic arm speed allows us to accomplish the task faster than keeping the speed to a constant value, furthermore the adaptive trajectory is safer and more comfortable from the operator point of view. As we expected, a small number of activations has a big impact in the efficiency for the adaptive system. This is particularly evident in the Place behavior, since interaction and precision are not requested, the task can be accomplished with minimal activations. The critical operations for the Safety and Reliability are the Give and Pick operations. As for safeness, we have observed that the Give interaction requires more care in HRI (where the robot has to pass an object to the operator) than the Receive one (where the robot has to receive an object from the operator) causing more frequent unsafe interactions.

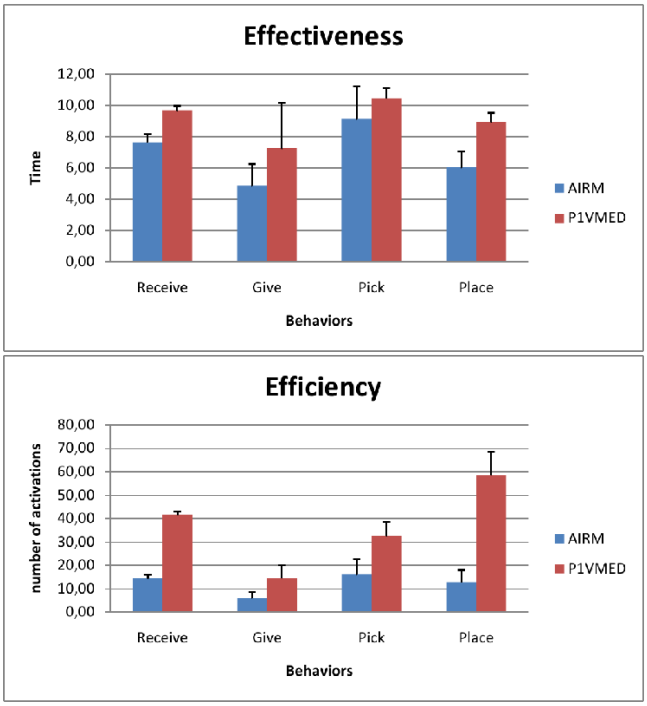


Figure 24: Effectiveness (time taken) and Efficiency (activations) evaluation criteria.

The same happens for reliability, indeed, passing an object to a human is more difficult than receiving an object. Although in these cases the success rate is not equal to 100% (as in the cases of Receive and Place behaviors), the architecture endowed with AIRMs seems more reliable than the P1Vmed standard architecture. For example, in the picking behavior the slower speed of the adaptive architectures permits a more accurate grip of the object.

In this section, we have illustrated a human-robot interactive system endowed with attention mechanisms used to coordinate simple manipulation tasks. In the proposed attentional model, each behavior is equipped with an adaptive clock and an updating policy that changes the frequency of sensory readings (focusing the attention towards relevant aspects of the external environment) and modulates the emergent behavior in terms of variations of the robot arm speed. We have defined a simple control architecture for HRI considering pick-and-place and give-and-receive attentional behaviors. To assess the system performance we have also introduced suitable evaluation criteria taking into account safety, reliability, efficiency, and effectiveness. The role of the attentional system is to find a trade-off between safety, effectiveness, and reliability in human-robot interaction and cooperation.

6 Conclusion

We have presented some concepts to build an interactive robot capable to share the workspace with humans and to become a companion or a co-worker. We have focused on the exchange of object between human and robot, which is a fundamental task for HRI. We have adressed firstly the architecture aspect and shown the importance of the communication between the different software modules. The grasp planning has then been studied to exhibit the importance of double grasp to exchange an object.

Then, we have presented how to plan and adapt robot motion to take into account the human and his/her behavior. The motion planner we have presented produces trajectories as series of cubic functions in joint or Cartesian space. This trajectory can then be adapted or modified to cope with the human changes in real time.

The last section has presented an attentional mechanism used to coordinate manipulation tasks. We have shown its interest to trade off between safety, effectiveness, and reliability in HRI and cooperation.

Of course, this approach of HRI is not complete and lots of points still need to be investigated like the exchange of information, the manipulation of more complex objects with two hands or the accomplishment of more complex tasks. For example, for a robot and a human intuitively exchange an object, they must exchange tactile information and the robot must be capable to generate and understand this information. This point is a challenge to build reliable and efficient robots that pick-and-give or receive-and-place objects in industrial environment. So, we can see that to realize simple daily tasks in interaction with human, robots need a lot of functionalities from human attention systems and supervision to tactile dialog and control. But such a robot cannot only share the space safely with humans but also do tasks for humans or help humans do tasks in an intuitive way. In this sense, the main result of this work is to have demonstrated the possibility to build intuitive and safe manipulator robots.

7 acknowledgement

The research leading to these results has been supported by the DEXMART Largescale integrating project, which has received funding from the European Communitys Seventh Framework Programme (FP7/2007-2013) under grant agreement ICT- 216239. The authors are solely responsible for its content. It does not represent the opinion of the European Community and the Community is not responsible for any use that might be made of the information contained therein.

The authors would like to thank Jean-Philippe Saut for his participation to the grasp planner, Emrah Akin Sisbot for the human aware planner, Mokhtar Gharbi for the manipulation planner, Matthieu Herrb and Anthony Mallet for their help to develop software and hardware components and all others.

References

[1] Alami, R., Albu-Schaeffer, A., Bicchi, A., Bischoff, R., Chatila, R., De Luca, A., De Santis, A., Giralt, G., Guiochet, J., Hirzinger, G., Lippiello, V., Mattone, R., Sen, S., Siciliano, B., Tonietti, G., Villani, L.: Safe and dependable physical human-robot interaction in anthropic domains: State of the art and challenges. In: Procceedings IROS Workshop on Physical Human-Robot interaction. Beijing, China (2006)

[2] Arbib, M.A.: Schema theory. In: The handbook of brain theory and neural networks, pp. 830–834. MIT Press, Cambridge, MA, USA (1998)

[3] Berchtold, S., Glavina, B.: A scalable optimizer for automatically generated manipulator motions. In: IEEE/RSJ Int. Conf. on Intel. Rob. And Sys. (1994)

[4] Berenson, D., Diankov, R., Nishiwaki, K., Kagami, S., Kuffner, J.: Grasp planning in complex scenes. In: IEEE-RAS International Conference on Humanoid Robots (Humanoids07) (2007)

[5] Bicchi, A.: On the closure properties of robotic grasping. Int. J. Rob. Res. **14**(4), 319–334 (1995). DOI http://dx.doi.org/10.1177/027836499501400402

[6] Bicchi, A., Tonietti, G.: Fast and soft arm tactics: Dealing with the safety-performance trade-off in robot arms design and control. Robotics and Automation Magazine pp. 22–33 (2004)

[7] Bounab, B., Sidobre, D., Zaatri, A.: Central axis approach for computing n-finger force-closure grasps. Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on pp. 1169–1174 (2008). DOI 10.1109/ROBOT.2008.4543362

[8] Breazeal, C.: Designing sociable robots. MIT Press, Cambridge, MA, USA (2002)

[9] Brock, O., Khatib, O.: Real-time re-planning in high-dimensional configuration spaces using sets of homotopic paths. In: IEEE Int. Conf. Robot. And Autom. (2000)

[10] Broquère, X.: Planification de trajectoire pour la manipulation d'objets et l'interaction homme-robot. Ph.D. thesis, LAAS-CNRS and Université de Toulouse, Paul Sabatier (2011)

[11] Broquère, X., Sidobre, D.: From motion planning to trajectory control with bounded jerk for service manipulator robots. In: IEEE Int. Conf. Robot. And Autom. (2010)

[12] Broquère, X., Sidobre, D., Herrera-Aguilar, I.: Soft motion trajectory planner for service manipulator robot. In: IEEE/RSJ Int. Conf. on Intel. Rob. And Sys. (2008)

[13] Burattini, E., Finzi, A., Rossi, S., Staffa, M.: Attentive monitoring strategies in a behavior-based robotic system: an evolutionary approach. Proceedings of the International Conference on Emerging Security Technologies (EST2010) pp. 153 – 158 (2010)

[14] Burattini, E., Finzi, A., Rossi, S., Staffa, M.: Attentional human-robot interaction in simple manipulation tasks. In: Proceedings of HRI-2012 (to appear) (2012)

[15] Burattini, E., Rossi, S.: A robotic architecture with innate releasing mechanism. In: 2nd International Symposium on Brain, Vision and Artificial Intelligence, *Lecture Notes in Computer Science*, vol. 4729, pp. 576–585. Springer (2007)

[16] Burattini, E., Rossi, S.: Periodic adaptive activation of behaviors in robotic system. Int. J. Pattern Recognition and Artificial Intelligence - Special Issue on Brain, Vision and Artificial Intelligence **22**(5), 987–999 (2008)

[17] Cakmak, M., Srinivasa, S.S., Lee, M.K., Forlizzi, J., Kiesler, S.: Human preferences for robot-human hand-over configurations. In: Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on, pp. 1986 –1993 (2011). DOI 10.1109/IROS.2011.6094735

[18] Cakmak, M., Srinivasa, S.S., Lee, M.K., Kiesler, S., Forlizzi, J.: Using spatial and temporal contrast for fluent robot-human hand-overs. In: Proceedings of the 6th international conference on Human-robot interaction, HRI '11, pp. 489–496. ACM, New York, NY, USA (2011). DOI http://doi.acm.org/10.1145/1957656.1957823. URL http://doi.acm.org/10.1145/1957656.1957823

[19] Carbone, A., Finzi, A., Orlandini, A., F.Pirri: Model-based control architecture for attentive robots in rescue scenarios. Auton. Robots **24**(1), 87–120 (2008)

[20] Ciocarlie, M., Goldfeder, C., Allen, P.: Dimensionality reduction for hand-independent dexterous robotic grasping. Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on pp. 3270–3275 (2007). DOI 10.1109/IROS.2007.4399227

[21] Cooper, R., Shallice, T.: Contention scheduling and the control of routine activities. Cognitive Neuropsychology **17**, 297–338 (2000)

[22] Cornelà, J., Suárez, R.: Determining independent grasp regions on 2d discrete objects. Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on pp. 2941–2946 (2005). DOI 10.1109/IROS.2005.1545422

[23] Cornellà, J., Suárez, R.: Fast and flexible determination of force-closure independent regions to grasp polygonal objects. Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on pp. 766–771 (2005)

[24] Cortés, J., Siméon, T.: Sampling-based motion planning under kinematic loop-closure constraints. Utrecht, Netherland (2004)

[25] Diankov, R., Ratliff, N., Ferguson, D., Srinivasa, S., Kuffner, J.: Bispace planning: Concurrent multi-space exploration. In: Proceedings of Robotics: Science and Systems IV. Zurich, Switzerland (2008)

[26] Ding, D., Liu, Y.H., Wang, S.: The synthesis of 3d form-closure grasps. Robotics and Automation, 2000. Proceedings. ICRA '00. IEEE International Conference on **4**, 3579–3584 vol.4 (2000). DOI 10.1109/ROBOT.2000.845289

[27] Edsinger, A., Kemp, C.C.: Human-robot interaction for cooperative manipulation: Handing objects to one another. In: RO-MAN 2007, pp. 1167–1172 (2007)

[28] Ferguson, D., Stentz, A.: Anytime rrts. In: IEEE/RSJ Int. Conf. on Intel. Rob. And Sys. (2006)

[29] Ferrari, C., Canny, J.: Planning optimal grasps. Robotics and Automation, 1992. Proceedings., 1992 IEEE International Conference on pp. 2290–2295 vol.3 (1992). DOI 10.1109/ROBOT.1992.219918

[30] Flash, T., Hogan, N.: The coordination of arm movements: an experimentally confirmed mathematical model. Journal of neuroscience (1985)

[31] Frintrop, S., Jensfelt, P., Christensen, H.I.: Attentional landmark selection for visual slam. In: Proc. of IROS 2006 (2006)

[32] Goldfeder, C., Allen, P., Lackner, C., Pelossof, R.: Grasp planning via decomposition trees. Robotics and Automation, 2007 IEEE International Conference on pp. 4679–4684 (2007). DOI 10.1109/ROBOT.2007.364200

[33] Haddadin, S., Albu-Schäffer, A., Hirzinger, G.: Requirements for Safe Robots: Measurements, Analysis and New Insights. International Journal of Robotics Research

[34] Hall, E.T.: A system for the notation of proxemic behavior. American anthropologist (1963)

[35] Han, L., Amato, N.: A kinematics-based probabilistic roadmap method for closed chain systems. pp. 233–246. A K Peters, Wellesley, MA (2001)

[36] Harada, K., Kaneko, K., Kanehiro, F.: Fast grasp planning for hand/arm systems based on convex model. Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on pp. 1162–1168 (2008). DOI 10.1109/ROBOT.2008.4543361

[37] Huber, M., Rickert, M., Knoll, A., Brandt, T., Glasauer, S.: Human-robot interaction in handing-over tasks. In: ROMAN (2008)

[38] Huebner, K., Ruthotto, S., Kragic, D.: Minimum volume bounding box decomposition for shape approximation in robot grasping. Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on pp. 1628–1633 (2008). DOI 10.1109/ROBOT.2008.4543434

[39] Ikuta, K., Ishii, H., Nokata, M.: Safety Evaluation Method of Design and Control for Human-Care Robots. International Journal of Robotics Research (2003)

[40] Jaillet, L., Cortés, J., Siméon, T.: Sampling-based path planning on configuration-space costmaps. IEEE Transactions on Robotics (2010)

[41] Kahneman, D.: Attention and Effort. Englewood Cliffs, NJ: Prentice-Hall (1973)

[42] Koay, K.L., Sisbot, E.A., Syrdal, D.A., Walters, M.L., Dautenhahn, K., Alami, R.: Exploratory study of a robot approaching a person in the context of handling over an object. In: Association for the Advancement of Artificial Intelligence Spring Symposia, AAAI. Palo Alto, CA, USA (2007)

[43] LaValle, S.M., Kuffner, J.: Rapidly-exploring random trees: Progress and prospects. In: Workshop on the Algorithmic Foundations of Robotics (2001)

[44] LaValle, S.M., Kuffner, J.J.: Rapidly-exploring random trees: Progress and prospects. In: Proceedings of Workshop on the Algorithmic Foundations of Robotics (2000)

[45] Mainprice, J., Sisbot, E., Jaillet, L., Cortés, J., Siméon, T., Alami, R.: Planning Human-aware motions using a sampling-based costmap planner. In: IEEE Int. Conf. Robot. And Autom. (2011)

[46] Marler, R., Rahmatalla, S., Shanahan, M., Abdel-Malek, K.: A new discomfort function for optimization-based posture prediction (2005)

[47] Miller, A., Allen, P.: Graspit! a versatile simulator for robotic grasping. Robotics & Automation Magazine, IEEE **11**(4), 110–122 (2004). DOI 10.1109/MRA.2004.1371616

[48] Miller, A., Knoop, S., Christensen, H., Allen, P.: Automatic grasp planning using shape primitives. Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on **2**, 1824–1829 vol.2 (2003)

[49] Mitsunaga, N., Asada, M.: Visual attention control for a legged mobile robot based on information criterion. In: Proc. of IROS-2002, pp. 244–249 (2002)

[50] Nagai, Y., Hosoda, K., Morita, A., Asada, M.: A constructive model for the development of joint attention. Connect. Sci. **15**(4), 211–229 (2003)

[51] Nguyen, V.D.: Constructing force-closure grasps. Robotics and Automation. Proceedings. 1986 IEEE International Conference on **3**, 1368–1373 (1986)

[52] Nonaka, S., Inoue, K., Arai, T., Mae, Y.: Evaluation of human sense of security for coexisting robots using virtual reality. 1st report: evaluation of pick and place motion of humanoid robots. In: IEEE Int. Conf. Robot. And Autom. New Orleans, USA (2004)

[53] Norman, D., Shallice, T.: Attention in action: willed and automatic control of behaviour. Consciousness and Self-regulation: advances in research and theory **4**, 1–18 (1986)

[54] Ott, C., Eiberger, O., Friedl, W., Bauml, B., Hillenbrand, U., Borst, C., Albu-Schaffer, A., Brunner, B., Hirschmuller, H., Kielhofer, S., et al.: A Humanoid Two-Arm System for Dexterous Manipulation. In: Humanoid Robots, 2006 6th IEEE-RAS International Conference on, pp. 276–283 (2006)

[55] Ponce, J., Sullivan, S., Sudsang, A., Boissonnat, J.D., Merlet, J.P.: On computing four-finger equilibrium and force-closure grasps of polyhedral objects. Int. J. Rob. Res. **16**(1), 11–35 (1997). DOI http://dx.doi.org/10.1177/027836499701600102

[56] Roa, M., Suárez, R.: Independent contact regions for frictional grasps on 3d objects. Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on pp. 1622–1627 (2008). DOI 10.1109/ROBOT.2008.4543433

[57] Santello, M., Flanders, M., Soechting, J.F.: Postural hand synergies for tool use. J. Neurosci. **18**(23), 10,105–10,115 (1998). URL http://www.jneurosci.org/cgi/content/abstract/18/23/10105

[58] Saxena, A., Driemeyer, J., Ng, A.Y.: Robotic grasping of novel objects using vision. International Journal of Robotics Research **27**(2), 157–173 (2008). DOI http://dx.doi.org/10.1177/0278364907087172

[59] Sisbot, E.A., Clodic, A., Alami, R., Ransan, M.: Supervision and motion planning for a mobile manipulator interacting with humans. In: Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction, HRI '08, pp. 327–334 (2008)

[60] Sisbot, E.A., Marin-Urias, L.F., Alami, R., Siméon, T.: Human aware mobile robot motion planner. IEEE Transactions on Robotics (2007)

[61] Sisbot, E.A., Marin-Urias, L.F., Broquère, X., Sidobre, D., Alami, R.: Synthesizing robot motions adapted to human presence - a planning and control framework for safe and socially acceptable robot motions. I. J. Social Robotics **2**(3), 329–343 (2010)

[62] Storn R., P.K.: Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces. Journal of Global Optimization **11**, 341–359 (1997)

[63] Suarez, R., Roa, M., Cornella, J.: Grasp quality measures. In: Universitat Politecnica de Catalunya (UPC), Technical Report (2006)

[64] Tinbergen, N.: The study of instinct. Oxford University press (1951)

[65] Todorov, E., Ghahramani, Z.: Analysis of the synergies underlying complex hand manipulation. Engineering in Medicine and Biology Society, 2004. IEMBS '04. 26th Annual International Conference of the IEEE **2**, 4637–4640 (2004). DOI 10.1109/IEMBS.2004.1404285

[66] Xue, Z., Marius Zoellner, J., Dillmann, R.: Grasp planning: Find the contact points. Robotics and Biomimetics, 2007. ROBIO 2007. IEEE International Conference on pp. 835–840 (2007). DOI 10.1109/ROBIO.2007.4522271

[67] Yershova, A., LaValle, S.: Deterministic sampling methods for spheres and so(3). In: IEEE International Conference on Robotics and Automation (2004)

[68] Zinn, M., Khatib, O., Roth, B., Salisbury, J.K.: Playing it safe [human-friendly robots]. IEEE Robotics & Automation Magazine (2004)

[69] Zucker, M., Kuffner, J., Branicky, M.: Multipartite rrts for rapid replanning in dynamic environments. In: Robotics and Automation, 2007 IEEE International Conference on (2007)