Cooperative Vision-Based Object Transportation by Two Humanoid Robots in a Cluttered Environment

Antoine Rioux, Claudia Esteves, Jean-Bernard Hayet, Wael Suleiman

International Journal of Humanoid Robotics

Cooperative Vision-Based Object Transportation by Two Humanoid Robots in a Cluttered Environment

Antoine Rioux

Electrical and Computer Engineering Department, Faculty of Engineering, University of Sherbrooke, Canada
antoine.rioux@usherbrooke.ca

Claudia Esteves

Department of Mathematics. Universidad de Guanajuato, México
cesteves@cimat.mx

Jean-Bernard Hayet

Centro de Investigación en Matemáticas (CIMAT), Guanajuato, México
jbhayet@cimat.mx

Wael Suleiman

Electrical and Computer Engineering Department, Faculty of Engineering, University of Sherbrooke, Canada
wael.suleiman@usherbrooke.ca

Received Day Month Year
Revised Day Month Year
Accepted Day Month Year

Corresponding Author (Wael Suleiman)

Although in recent years there have been quite a few studies aimed at the navigation of robots in cluttered environments, few of these have addressed the problem of robots navigating while moving a large or heavy object. Such a functionality is especially useful when transporting objects of different shapes and weights without having to modify the robot hardware.

In this work, we tackle the problem of making two humanoid robots navigate in a cluttered environment while transporting a very large object that simply could not be moved by a single robot. We present a complete navigation scheme, from the incremental construction of a map of the environment and the computation of collision-free trajectories to the design of the control to execute those trajectories. We present experiments made on real Nao robots, equipped with RGB-D sensors mounted on their heads, moving an object around obstacles. Our experiments show that a significantly large object can be transported without modifying the robot main hardware, and therefore that our scheme enhances the humanoid robots capacities in real-life situations.

Our contributions are: (1) a low-dimension multi-robot motion planning algorithm that finds an obstacle-free trajectory, by using the constructed map of the environment as an input, (2) a framework that produces continuous and consistent odometry data, by fusing the visual and the robot odometry information, (3) a synchronization system that uses the projection of the robots based on their hands positions coupled with the visual

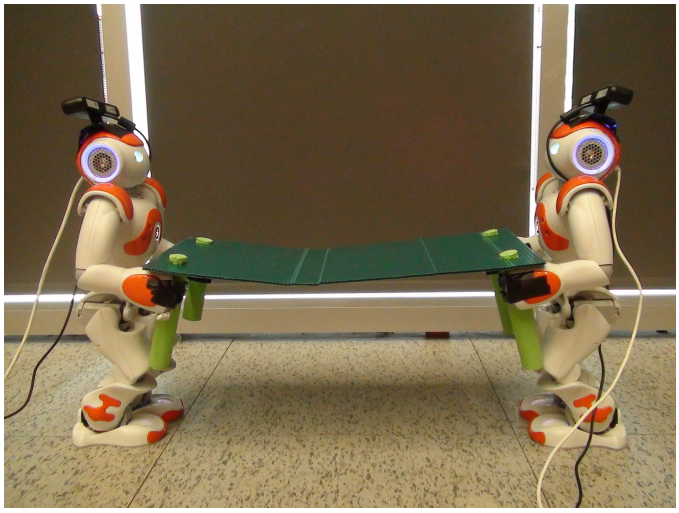


Fig. 1. Two Nao robots holding an object together. We propose a motion planning strategy for the Nao-object-Nao system to navigate in a cluttered environment and a synchronization approach that allows the motions done by the two Naos without compromising the whole system stability because of the robots swinging movement.

feedback error computed from a frontal camera, (4) an efficient real-time whole-body control scheme that controls the motions of the closed-loop robot-object-robot system.

Keywords: Navigation; Collaborative tasks; SLAM; Whole-body control; Motion planning.

1. Introduction

The main motivation behind developing robots with arms is that they can manipulate loads. This ability can be useful for a wide range of actions, including transporting objects from one place to another. However, using one robot only, the maximum payload is generally low and the size of the transported objects is necessarily limited. One way to deal with this issue is to distribute the weight or the large surface of the object to be carried among multiple robots, as humans do. In this work, we deal with the specific problem of having two humanoid robots cooperating to maneuver a bulky object among obstacles, as illustrated in Fig. 1.

Many studies have been done on robots moving objects to a specific goal [1]. However, most of them are executed with multiple wheeled robots that position themselves around the object to push it in the desired direction [2,3,4,5,6]. In the aforementioned works, the manipulator comes in contact with the object at only one point, which barely allows any control over the object while moving and manipulating it. Holonomic wheeled robots are much less complex to control than humanoid robots and are mainly used here to slide box-like objects on the ground. Robots with humanoid arms have a better control on the structure of an object

and are, therefore, more suitable to control objects within a large range of different shapes.

More advanced and specialized models of wheeled robots can possess humanoid torsos and arms, such as the humanoid ARMAR robots [7], which allow them to keep a high level of control when holding/transporting objects or interacting in real-time with the environment and humans [8,9,10,11,12]. However, most environments made by and for humans are more suitable for fully humanoid robots. To this end, the subject of transporting bulky or heavy objects with a humanoid robot has received attention in the past years, and many different techniques have been developed. For instance, a human-humanoid co-working framework to transport a table has been proposed in [13], lifting the load from the ground [14,15,16] or using a part of the transported object as a pivot to move it [17,18,19].

Other works have explored the use of humanoid robots cooperating to transport an object. One of the most popular control schemes for cooperation with multiple robots, be the robots big or small, is the leader-follower control scheme. One of the robots, the leader, based on its position and its surrounding, computes the common plan of the system or is being directly controlled by a human operator. The second robot, the follower, simply follows the leader robot. While relatively simple to implement, this technique has many flaws when used in closed-loop cooperation, which is why other works have used disjoint objects [20] or have constrained the robots motions to one axis [21]. Since the follower only responds to the leader's movement after it has already started, a significant time delay is introduced. Moreover, interpretation errors of the leader's movements can destabilize the closed-loop system and can cause unexpected falling.

Another popular control scheme for this problem is to use a set of synchronized controllers. With this technique, there is no apparent master robot and an external centralized controller manages all the robots simultaneously, based on the complete information provided by the robots. As a result, the synchronized robots start, move and stop together. This way, the system is highly responsive and any error is shared within the whole system. To achieve this, one can model the entire robot-object-robot system as a quadruped with a rigid body, thus preventing the robots to manipulate the object freely [22]. The dynamic of the system is simplified, and many degrees of freedom can be removed by adding virtual constraints. The work of [23] explored this strategy with the human-sized HRP2 robot. Even-though their results look promising in simulation, their implementation does not consider navigation among obstacles nor the robot localization and mapping. It also makes an extensive use of expensive six-axis force sensors located in the wrist, which makes the approach difficult to generalize to many affordable robots that are not equipped with such force sensors.

In a preliminary version of this work [24], we proposed a framework for cooperative humanoid robots navigation in a cluttered environment. The main limitation was that the distance between the robots was not monitored, which could yield to the destabilization of the robots during the motion. In the work presented here, we

present a remedy to this problem.

In this research, we assume that the robots are sharing some information such as the mapping of the environment and the desired trajectory to follow. The proposed framework can be executed on each robot, however as the Nao robots have very limited computational capabilities in comparison with other high-end humanoid robots that have often multiple onboard computers, we execute a part of the framework on an external PC, such as the desired trajectory, the environment mapping with the SLAM system and the visual feedback.The results are then transmitted to the robots via Ethernet connexion.

The main contribution of this work is to provide a complete framework for cooperative autonomous humanoid robots navigations in a cluttered environment, while manipulating an object in a closed kinematic chain, with no leader-follower approach, but with a synchronized execution of a centrally-decided trajectory. An improved odometry data fusion scheme in presence of unreliable real-time SLAM information is detailed. A synchronization mechanism, which uses the arms, trajectories, robots reflection positions and relative visual positions is also proposed. In this paper, we also introduce an adapted visual feedback technique, which has greatly improved the results and the dynamic stability of the robots w.r.t. [24].

To address the problem of making a closed-loop robot-object-robot navigate in a cluttered environment, the following sub-problems need be solved: (I) planning, (II) controlling the object, (III) sensing the environment, and (IV) synchronizing the system. This paper is organized according to these sub-problems. Section 2 presents an anytime search-based planner that exploits a given set of motion primitives. This planner considers both robots and the object footprint in order to plan a safe trajectory between obstacles. In Section 3, we show how real-time information from a consumer-level depth camera allows us to perform simultaneous localization and mapping (SLAM) of the surrounding obstacles in the cluttered environment. Section 4 details the different synchronization mechanisms and the way they are integrated together. Section 5 describes how to determine the humanoid robots' footprints, and then how to compute the robots' feet and hands trajectories in order to minimize the swing effect, synchronize the motions of the two robots, and follow the planned trajectory by using a task priority whole-body control scheme. Finally, in Section 6, results of simulations and real world experiments are presented and discussed.

2. Planning a Valid Path

To navigate through a cluttered environment, the computation of a collision-free path for both robots is essential. For this, we chose a lattice-based graph planning with an ARA* search [25], motivated by the use of a collection of motion primitives ensuring feasible robot-object-robot configurations and transitions. The environment is modelled as a 2D grid cost-map that discriminates obstacles from free space by using a fixed threshold on the cost value. Moreover, it allows obstacles inflations to

increase the security margin (see Section 2.4).

2.1. *State Representation*

Each node of the search graph needs a representation of a full state including the configurations of both robots and of the object in the plane. To achieve this, it is possible to model the state in $\mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{R}^2 \times \mathbb{S}^1$:

$$s = (x_{r1}, y_{r1}, \theta_{r1}, x_{ob}, y_{ob}, \theta_{ob}, x_{r2}, y_{r2}, \theta_{r2}), \tag{1}$$

where x_{ri}, y_{ri} and θ_{ri} $(i = 1, 2)$ are the positions and orientation of the pelvis of the i-th robot, and x_{ob}, y_{ob} and θ_{ob} are those of the object. Note that these simplified, planar models for the robots and the object are used for the planning step only.

As the working space of our robot's arms is too small to fully take advantage of both rotation and translation, the system is simplified by setting a pivot point at the middle of the pair of hands for both robots, as shown in Fig. 2. The closed-loop grasping of the robot on the table is shown in Fig. 1. The pivot points positions have been chosen to maximize the rotation range within the robot workspace, resulting in a smaller 5 dimensions state space $\mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{S}^1 \times \mathbb{S}^1$:

$$s = (x_{r1}, y_{r1}, \theta_{r1}, \theta_{ob}, \theta_{r2}). \tag{2}$$

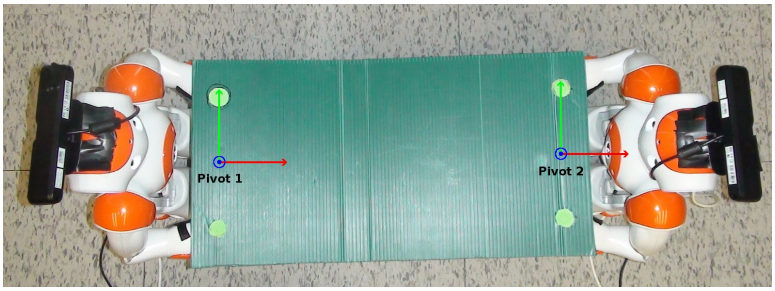


Fig. 2. Top view: Pivots position

Even though the above simplification removes the ability of the object to translate on the plane independently, the robot retains enough manipulability to minimize the robot-object-robot collision area around obstacles. The simplified state representation of equation (2) is shown in Fig. 3.

2.2. *Transition Model*

In a lattice-based graph planner, transitions between nodes are triggered by actions chosen within a finite fixed-set of motion primitives. Motion primitives allow the decomposition of complex motion generation in robotics and it is very likely that they are also used by humans and animals [26]. An important feature of methods

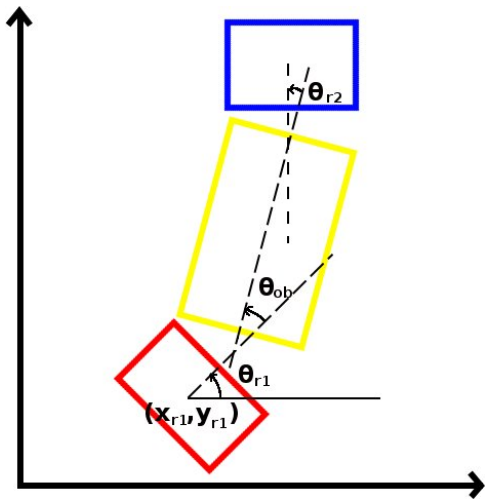


Fig. 3. Simplified state representation. In red and blue: the two humanoid robots. In yellow: the load.

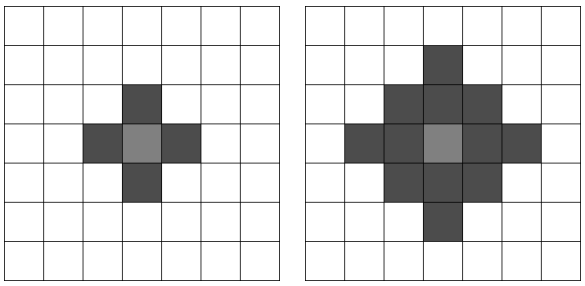
based on the lattice representation is that all connections are feasible paths. The way the expansion of nodes is done during the search encodes the aforementioned notion of motion primitive. An example of the first two expansions of different graph search methods is shown in Fig. 4. The representation we chose (illustrated by the two expansion images on the right) is really suitable for highly constrained systems, such as a system of two robots transporting an object, in contrast to other commonly used forms of encoding transitions in graph search, including Von Neumann or Moore neighborhoods (illustrated by the expansion images on the left).

The full set of motion primitives used for this problem is shown in Fig. 5. It includes (a) forward, backward, sideways motion and every diagonal motion, (b) rotations around each robot and around the object center, and special movements such as (c) C-turns and (d) S-turns. The last two are more complex and make use of the hands articulations to increase agility around obstacles. Executing complex motions specific to a system in a coherent and logical way is the main reason we use motion primitives, e.g., over a more traditional method doing a homogeneous sampling of the system DOFs. The joints of the system that allow these particular motions are the aforementioned pivot points.

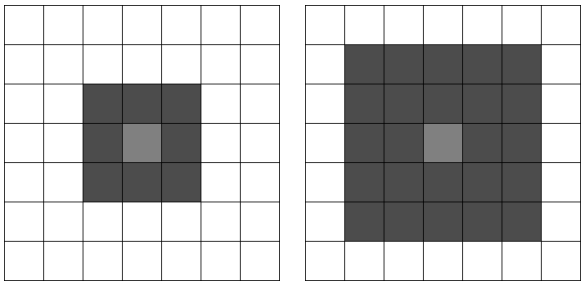
2.3. *Path Cost Function*

The cost of a transition from state s to $s' \in \mathbb{R}^2 \times \mathbb{S}^3$ is based on the time to execute that transition and is computed as follows:

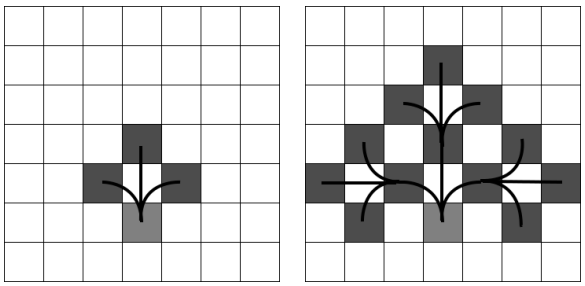
$$g(s,s') = \begin{cases} \frac{\sqrt{(\Delta x_{r1})^2+(\Delta y_{r1})^2}}{\dot{\boldsymbol{r_1}}^+} \times DF & \text{if } \Delta x_{r1} \neq 0 \text{ or } \Delta y_{r1} \neq 0 \\ \frac{\sqrt{(\Delta x_{r2})^2+(\Delta y_{r2})^2}}{\dot{\boldsymbol{r_2}}^+} \times DF & \text{otherwise} \end{cases} \tag{3}$$



(a) Von Neumann neighborhood 1st & 2nd expansions



(b) Moore neighborhood 1st & 2nd expansions

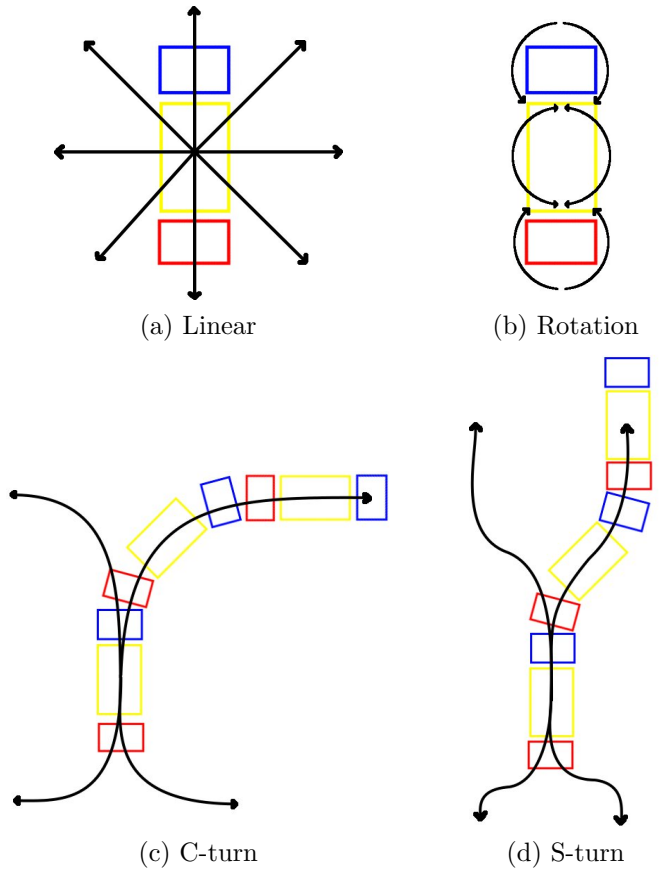


(c) examples from our set of motion primitives 1st & 2nd expansions

Fig. 4. Different forms of graph search methods.

where Δx_{ri}, Δy_{ri} are the variations of the x and y coordinates of the i-th robot pelvis, between states s and s', DF is a difficulty factor associated with each primitive, and $\dot{\boldsymbol{r}}_{\boldsymbol{i}}{}^{+}$ is the i-th robot maximal linear velocity. This ratio gives us the approximate time to execute the primitive. To avoid collision with obstacles and to keep a safety distance to those obstacles, transitions close to obstacles have higher costs. Note that only one primitive is selected to connect two states s and s'.

The difficulty factor DF is a special value set by the system expert in order to prioritize or penalize certain motions, which results in a smoother and a more natural looking trajectory. For instance, turning in place then moving forward (Fig.



(a) Linear (b) Rotation

(c) C-turn (d) S-turn

Fig. 5. Set of possible motion primitives.

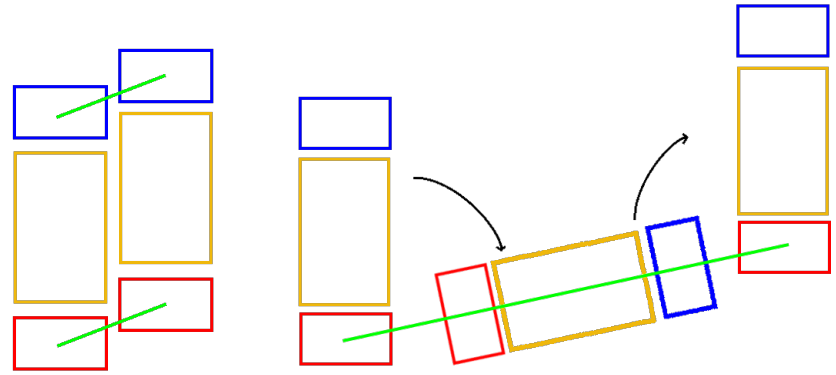
6 a) takes a longer time than moving in diagonal (Fig. 6 b). However, on a long distance, the former reduces the trajectory footprint and is therefore more natural looking while reducing the chances of drifts caused by the table movements. For those reasons, moving sideways has a higher DF than turning and moving forward. In sum, for small distances, the time cost takes over the DF; however, for long distances, it is more likely that turning in place and moving forward would be preferred. Examples of DF values are given in Table 1.

2.4. *Search Algorithm*

A* is one of the most popular search algorithms. In addition to the use of a path cost function, a heuristic biases the search towards the most promising states. Even though the solution found by A* is usually the optimal one, that solution does not always exist or may not be found within a reasonable time. The Anytime Repairing A* (ARA*) focuses on delivering a suboptimal solution as fast as possible; this

Primitive set	DF factor
Forward	1
Turn in place	2
Backward	3
Sideways	2
Diagonal	1

Table 1. The DF factor for different motion classes.



(a) Short diagonal (b) Long diagonal

Fig. 6. Examples of the effect of the DF.

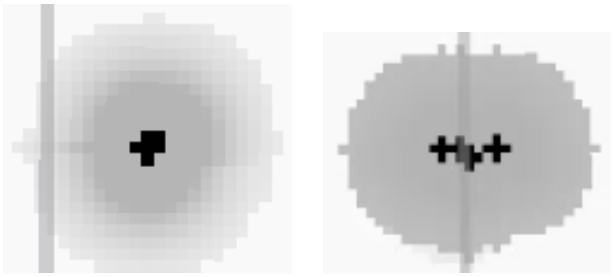
solution is then optimized iteratively within a predefined limited time. Also, the states are expanded from goal to start, so that the heuristic costs remain valid when replanning and do not need to be recomputed. The cost function takes the form of:

$$f(s, s') = g(s, s') * \max_{\sigma \in \pi(s,s')} (Cost_{cell}(\sigma)) + \epsilon h(s'), \epsilon \geq 1 \tag{4}$$

where $\pi(s, s')$ is the path connecting s and s', σ is a cell along this path, $g(s, s')$ is the path cost of equation (3), $h(s')$ is the heuristic that uses a 2D grid containing all the Dijkstra distances from the goal to the start states and

$$Cost_{cell}(\sigma) = \begin{cases} 1 & \text{free space} \\ 2 \text{ to } 99 & \text{inflation} \\ \infty & \text{obstacles} \end{cases} \tag{5}$$

which is the cost of cell σ that intersects the path connecting s and s'. The search is biased towards states closer to goal (because of the term $h(s')$) and returns a solution that is, at worst, ϵ times the cost of the optimal solution.



(a) Gradient (b) Solid

Fig. 7. Types of inflation.

The inflation in Eq. 5 is a zone around obstacles where all the cells have a higher cost. It is used as a security margin to bias the search farther from obstacles and reduce the likelihood of collisions. It can be set as a decreasing gradient from the obstacles (See Fig. 7.a) or as a fixed value in the range above (See Fig. 7.b).

3. Simultaneous Localization and Mapping

To move in a cluttered environment, a robust and precise sensing input is primordial to determine the position of the obstacles, to detect collisions and to plan valid long-term and short-term paths. Also, the odometry drift must be constantly verified and corrected by an accurate localization mechanism to ensure that the planned path is closely followed. The human-sized humanoid robots, such as HRP-2 or the humanoid robots participating in the DARPA Challenge, are able to build a 3D map on the fly using their very sophisticated proprioceptive and exteroceptive sensors. However, the Nao robot has natively only two color cameras in its head for sensing and localization. A first approach would be to use those two cameras. However, this has proven to be a very difficult task [27,28]. Indeed, as explained previously, a humanoid robot swings laterally while walking. This effect coupled with low resolution cameras leads to pictures of poor quality. Furthermore, the field of view of the Nao cameras is greatly obstructed by the large object being transported and by the other robot situated right in front of it, in particular for the bottom camera. As a result, it is hard to determine precisely the position of the environment features and of the obstacles with respect to the robot only with the Nao's cameras.

3.1. *Real-Time Appearance-Based Mapping (RTAB-Map)*

We chose to add an RGB-D camera on the top of each Nao's head and to use it for mapping [29], based on the open source library RTAB-Map [30,31]. RTAB-Map is an RGB-D Graph-SLAM library that uses a bag-of-words technique for loop closure detection. A memory management system limits the quantity of information loaded in memory to ensure the constant satisfaction of large environment real-time

constraints. With these features, it can support large maps with kilometers-long paths and multi-sessions mapping and localization.

RTAB-Map also provides a robust odometry system based on visual information. It can create 3D maps of the environment as well as 2D occupancy grids by projecting the obstacles on the ground plane. In this work, we fuse the occupancy grids generated by both robots into a common one that is used for planning. Even-though our system could use probabilistic grids, the SLAM library that we use only provides deterministic occupancy grids at this moment.

3.2. *Odometry Fusion*

A problem that may occur with the visual odometry produced by RTAB-Map is that it may lose track of the position for multiple reasons, such as missing image features in the observed environment, rapid movements of the camera or intense oscillations. When this occurs, we could go back to where the tracking was lost, but this is not efficient and may even be impossible. Hence, a fusion of the visual odometry, the robot's internal odometry and the error between those two reference frames is used to improve the overall odometry.

Since the camera is rigidly linked to the robot by a transformation T_v^c, we can write $T_v^o = T_v^c * T_c^o$, where T_v^o, T_c^o are the homogeneous transformation matrices between the map frame (index o) and, respectively, the robot frame (index v) and the camera frame (index c). This equation can be rewritten to include the encoders-based robot odometry T_r^o (indice r), that does not take into account slipping, drift and other real world errors,

$$\begin{aligned} T_v^o &= T_v^c * T_c^o * T_r^{o-1} * T_r^o \\ &= T_v^r * T_r^o \end{aligned} \tag{6}$$

where T_v^r is the error between the encoders odometry and the visual odometry. Since this equation only holds while the visual odometry is valid, the last valid T_v^r, at time $t = t_{lost}$, is used when a loss of the visual odometry occurs at $t = t_{lost}$,

$$T_v^o(t) = \begin{cases} T_v^c(t) * T_c^r(t) * T_r^o(t) & \text{if } T_v^c \text{ is valid,} \\ T_v^r(t_{lost}) * T_r^o(t) & \text{otherwise.} \end{cases} \tag{7}$$

This approach consistently provides smooth odometry, assuming a constant error between the two odometries when the visual odometry fails. Even when the visual information is abruptly discontinued, it continues to generate sufficiently accurate localization data until an adequate image or a reset command is processed by RTAB-Map and the visual odometry is restored.

4. Synchronization

4.1. *Object Stability and Hand Stabilization*

During the transition between single and double support phases, the CoM moves horizontally from one support foot to the other [32]. This lateral (sway) motion causes the entire upper body to oscillate laterally at an amplitude proportional to the distance between the center of its feet, which, in our case, causes the transported object to move by the same amplitude.

Since the object to carry is fully controlled by the robots' hands, we could use this control to reduce the oscillating effect to improve the closed-loop kinematic chain stability. On the one hand, if both robots swing at the same time, synchronization is done easily, but the object and anything on it would swing dangerously. On the other hand, if the robots swing in any other way, the force generated by each robot movement will be transmitted to the other robot and may cause instability.

Our solution to compensate this instability without changing the walking gait is to use the robots' hands and keep them at a fixed position in space, *relatively to the planned trajectory*. This position is determined from the starting position of the robot and corresponds to the initial transformation between the robot feet and hands. The output of our corresponding hand stabilization system is $\dot{\boldsymbol{r}}_{lh}$, $\dot{\boldsymbol{r}}_{rh}$, which are the linear and angular velocity of the left and right hands respectively. Those outputs are integrated into the whole-body control scheme described in Section 5.

4.2. *Synchronized Reflections*

Now that the whole system is more stable with regard to the individual swing added by both robots, the position of each robot needs to be synchronized with the other one along the planned trajectory. To do so, the reflection of each robot with respect to the other is computed by using the robots respective hands, world frames and transported object properties. First, the position of the center of the object with respect to a robot i can be found by:

$$T_{ob}^{r_i} = midpoint(T_{h_r}^{r_i} * T_{ob}^{h_r}, T_{h_l}^{r_i} * T_{ob}^{h_l}) \tag{8}$$

where $T_{ob}^{r_i}$, $T_{h_{\{r,l\}}}^{r_i}$, $T_{ob}^{h_{\{r,l\}}}$ are respectively the current transformations between the robot i and the object frames, the robot i and its own hands frames and the object center and the robot i hands frames. The robot hand frame may be defined for each hand (hence the notation $T_{h_{\{r,l\}}}^{r_i}$). The function *midpoint* computes the midpoint of two transforms.

With these transformations defined, we set the reflection synchronization position for each robot as follows:

$$T_{p_i}^{o} = T_{r_j}^{o} * T_{ob}^{r_j} * {T_{ob}^{r_i}}^{-1} \qquad \text{for } i, j \in \{1, 2\} : i \neq j, \tag{9}$$

where $T_{p_i}^{o}$ is the reflected position of the i^{th} robot in the world frame and $T_{r_j}^{o}$ is the odometry data from the other robot j. Other points different from the center could

be used instead, such as the pivot points. However, a constant offset transformation would need to be taken into consideration in the previous equation. We can finally compare this reflected position to the actual position of each robot in order to determine the reflection synchronization error matrix e_{p_i}

$$e_{p_i} = T^o_{p_i} * T^{o\,-1}_{r_i} \tag{10}$$

This matrix error is then transformed to obtain the desired linear and angular velocity of the chest [33]. Since we are only interested in the horizontal velocities and angular velocity around the vertical axis for the chest frame, the velocity vector is truncated such as $\dot{\boldsymbol{r}}^s_{ch} \in \mathbb{R}^3$.

4.3. *Visual Feedback*

To further improve the synchronization, a second, visual technique, based on the optical flow observed from the robots cameras, is added. We have relied on the Lucas–Kanade method [34] to determine a sparse representation of the apparent motion between consecutive images of a video feed. However, in our case, since the robots need to be synchronized and should always remain close from one another, we used the optical flow between a reference image and the video feed received from each robot, in order for the robots to maintain a fixed relative position with respect to each other.

This underlying assumption simplifies the image processing problem: Instead of using a full image of the robot with the object and the (eventually cluttered) background, an ad-hoc mask is built to extract features in the reference image only on some relevant parts of the content, i.e., the robot. Note that the reference features are extracted only once, either offline or online at the initialization of the program, using the "good features to track" [35] (GFTT) to select the keypoints to track along the video frames. Another advantage is that the motion evaluated through the optical flow is absolute, that is, it represents directly the error of synchronization instead of accumulating image-to-image motions (and drift!) while keeping track of the errors. The mask and some extracted features are shown in Fig. 8.

As one of the assumption of optical flow methods is that the motion is small between the two images, the search for corresponding points is limited to a neighborhood of the last detection, which serves as the initial estimate. Since the search is focused in a small window, it is more robust to blur and reduces the number of bad matches in other regions of the robot or on the background, compared to more traditional features matching methods. This leads to less outliers, which in turn improves the speed and performance of most algorithms and geometrical verification (LMEDS, homography matrix computation and tests on reprojection errors). For points losses under occlusion, disturbance or rapid movements, we reset the estimate of the positions of these keypoints to the original mask features.

We transform the feature points in the reference image into approximate 3D coordinates in the camera frame at the reference position by

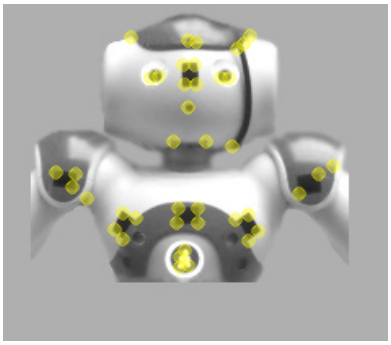


Fig. 8. The mask used for our visual feedback with the features extracted by GFTT.

$$\mathbf{P}_i = \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix} = d * K^{-1} * \begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} = d \begin{bmatrix} \frac{u_i - u_0}{f_u} \\ \frac{v_i - v_0}{f_v} \end{bmatrix} \tag{11}$$

where d is the (known) distance from the camera to the plane where the reference points are supposed to lie and $K = \begin{pmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}$ is the camera intrinsic parameters matrix.

Note that we use d and the equation 11 just once, so as to initialize the system and to determine the 3d coordinates of the points that will be tracked on the robot. This distance d (scale factor) is set manually on the initialization phase. After this initialization is done, the algorithm we apply is a planar form of the PnP problem.

When tracking the points, we evaluate at each frame the homography H between the 3D reference points $\mathbf{P}_i$ defined above and the tracked ones in frame t, $\mathbf{p}_i$, in a RANSAC scheme. Then we use the decomposition:

$$H \propto K \begin{bmatrix} R_1 & R_2 & t \end{bmatrix} \tag{12}$$

where R_1, R_2 are the first two columns of the 3D rotation matrix, and t the translation, to determine the camera localization [36]. From the orthonormality of these vectors, we get a first estimate as

$$\begin{bmatrix} R'_1 & R'_2 & t' \end{bmatrix} = \frac{K^{-1}}{\frac{1}{2}(\|R_1\| + \|R_2\|)} H \tag{13}$$

The third column R'_3 is found by taking the cross product between R'_1 and R'_2 to obtain the full 3D rotation matrix

$$R' = \begin{bmatrix} R'_1 & R'_2 & R'_1 \times R'_2 \end{bmatrix} \tag{14}$$

Finally, to get the closest rotation matrix to R', as suggested in [36], we perform a SVD decomposition on $R' = USV^T$ and set S to identity, to get

$$R_{SVD} = UV^T$$

Beforehand, the singular values of S are checked to verify they are close to one with a 10% tolerance. When they differ too much from one, it means that the rotation found is probably wrong and must be discarded. Using U and V, the homogeneous matrix expressing the robot position is then:

$$T_c^r = \begin{bmatrix} UV^T & t' \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \tag{15}$$

This gives us the relative positioning error between both robots. Similarly to Section 4.2, this error is transformed into a desired linear and angular velocity of the chest, and then truncated to obtain $\dot{\boldsymbol{r}}_{ch}^v \in \mathbb{R}^3$. By adding this visual feedback, as described in the next Section, the synchronization results have been greatly improved. Note that this velocity $\dot{\boldsymbol{r}}_{ch}^v$ computed from the optical flow is a relative one between the robots. Hence, when it is handled in the stack of tasks (see Section 5.1), the purpose is to equate the two robots absolute velocities. To allow this to happen, we divide the estimated velocity $\dot{\boldsymbol{r}}_{ch}^v$ in two equal halves, for each robot, so that the difference between the two is $\dot{\boldsymbol{r}}_{ch}^v$.

5. Control

Once a collision-free trajectory is found by the ARA* algorithm (see Section 2), a set of footprints are defined along the trajectory [37]. The second step is to define a Zero Moment Point (ZMP) trajectory and the robots footprints. Note that the robots footsteps are not directly synchronized. Our strategy is to plan the footprints of each robot, before executing the cooperative task, in function of the planned trajectory as shown in Fig. 9. All footsteps are then given a constant time to be executed. The footprints are however recomputed when a collision is foreseen or a drift from the planned trajectory is detected as the desired trajectory is deformed.

A trajectory of the Center of Mass (CoM) of the robot is then obtained using the preview control algorithm proposed in [32]. This algorithm, widely used in humanoid robotics, is simple to implement, yet efficient and yields a smooth CoM trajectory by minimizing the CoM jerk trajectory. The feet trajectories are obtained by spline interpolation between the footprints and the hands trajectories and orientations are defined in order to minimize the walking swing effect and, in our case, to follow the object orientation. The output of the locomotion algorithm are $\dot{\boldsymbol{r}}_c$, $\dot{\boldsymbol{r}}_{lf}$, $\dot{\boldsymbol{r}}_{rf}$, which are the computed linear and angular velocity of the CoM, left foot, right foot respectively.

5.1. *Hierarchy of Tasks*

To obtain the joint trajectories for each humanoid robot, a whole-body control scheme with prioritized tasks is formulated as follows:

$$
\begin{aligned}
&\min_{\dot{\boldsymbol{q}}} \dot{\boldsymbol{q}}^T \dot{\boldsymbol{q}} \\
&\text{subject to} \\
&\text{First priority (locomotion)} \quad \begin{cases} \boldsymbol{J}_c\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_c \\ \boldsymbol{J}_{lf}\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{lf} \\ \boldsymbol{J}_{rf}\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{rf} \end{cases} \\
&\text{Second priority (synchronization)} \quad \begin{cases} \boldsymbol{J}_{lh}\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{lh} \\ \boldsymbol{J}_{rh}\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{rh} \\ \boldsymbol{J}_p\, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_p \\ \boldsymbol{J}_{ch}\, \dot{\boldsymbol{q}} = \alpha_1\, \dot{\boldsymbol{r}}^s_{ch} + \alpha_2\, \dot{\boldsymbol{r}}^v_{ch} \end{cases} \\
&\text{Joint velocity limits} \quad \hat{\dot{\boldsymbol{q}}}^- \leq \dot{\boldsymbol{q}} \leq \hat{\dot{\boldsymbol{q}}}^+
\end{aligned}
\tag{16}
$$

where $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is the joint velocity vector to determine, n is the number of degrees of freedom, $\boldsymbol{J}_c \in \mathbb{R}^{3\times n}$, $\boldsymbol{J}_{lf} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{rf} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{lh} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{rh} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_p \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{ch} \in \mathbb{R}^{3\times n}$ are the Jacobian matrices of the CoM, left foot, right foot, left hand, right hand, pelvis joint and chest, respectively. α_1 and α_2 are user-defined positive constants, such as $\alpha_1 + \alpha_2 = 1$. They allow to parameterize the task designed to follow a reference chest velocity between two reference velocities. The first one, $\dot{\boldsymbol{r}}^s{}_{,ch}$ comes from the synchronized reflection module developed in Section 4.2; The second one, $\dot{\boldsymbol{r}}^v_{ch}$, arises from the visual synchronization module described in Section 4.3. In this stack of tasks scheme, the higher priority is given to the locomotion (velocities of the CoM and feet), to ensure the robot stability, while the synchronization tasks (see Section 4) are given a lower priority.

$\hat{\dot{\boldsymbol{q}}}^-$ and $\hat{\dot{\boldsymbol{q}}}^+$ are generalized joint velocity limits defined as follows:

$$
\begin{aligned}
\hat{\dot{\boldsymbol{q}}}^+_j &= \begin{cases} \dot{q}^+_j \, \frac{(q^+_j - q_j) - \rho_s}{\rho_i - \rho_s} & \text{if } q^+_j - q_j \leq \rho_i, \\ \dot{q}^+_j & \text{otherwise,} \end{cases} \\
\hat{\dot{\boldsymbol{q}}}^-_j &= \begin{cases} \dot{q}^-_j \, \frac{(q_j - q^-_j) - \rho_s}{\rho_i - \rho_s} & \text{if } q_j - q^-_j \leq \rho_i, \\ \dot{q}^-_j & \text{otherwise,} \end{cases}
\end{aligned}
\tag{17}
$$

where $\hat{\dot{\boldsymbol{q}}}_j$ is the j-th element of the vector $\hat{\dot{\boldsymbol{q}}}$, q_j is the value of joint j, q^+_j and q^-_j are the upper and lower limits for the joint j, ρ_i and ρ_s are user-defined positive constants, q_i is called the interference distance. It can be easily proven that the equalities constraints in (17), not only yield a motion within the humanoid velocity limits, but also allow the joints limits to be respected with a safety margin equal to

ρ_s:

$$q_j^- + \rho_s \leq q_j \leq q_j^+ - \rho_s$$

Eq. (17) provides a compact and efficient way to deal with both velocity and joint limits, as originally proposed in [38].

The optimization problem (16) can be transformed into a standard Quadratic Programming (QP) problem [37,39], which can be solved in real-time by using an appropriate QP solver.

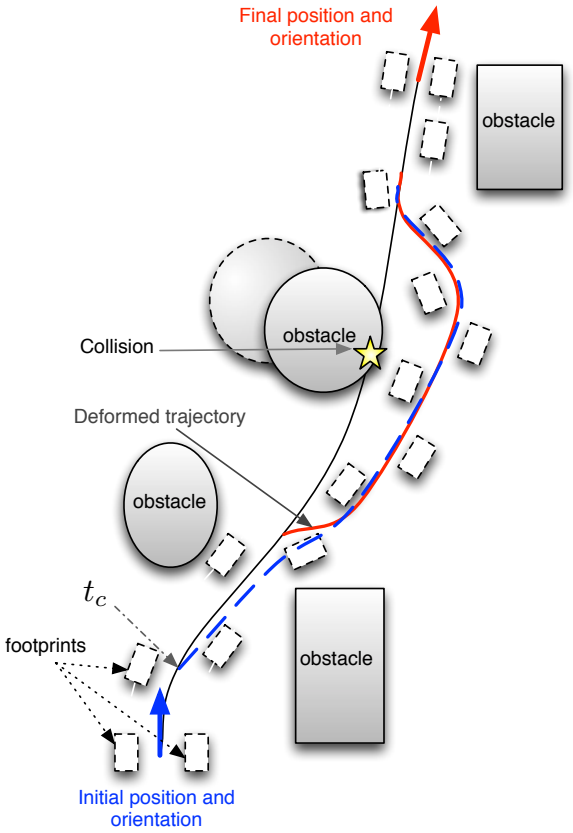


Fig. 9. Replanning in case of collision detection: t_c is the instant at which a collision is foreseen, the initial trajectory is in black, the new collision-free trajectory is in dashed-blue line, the deformed trajectory is in red line.

5.2. *Dynamic Collision Avoidance and Replanning*

It is important to check frequently for collision in case an obstacle has moved from its original position or the robots drift away from their planned trajectory. Hence, the

future trajectory is always monitored for potential collisions with obstacles in the 2D occupancy grid. When a collision is foreseen, a replanning step is necessary and the trajectory is deformed as shown, for the purpose of clarity for a single robot, in Fig. 9. Note that even though, at first glance, the support polygon seems increased by including the robot-object-robot closed loop, the robots' support polygons are still defined here individually by the contact between the feet and the ground. This is because the robots' arms are not fully bended, therefore the robots could fall forwards or backwards.

The new collision-free trajectory is found by the ARA* algorithm, which is applied from the goal to the point at which the collision has been predicted. If the potential collision is due to drift and the environment has not changed, the Dijkstra grid does not need to be recalculated, therefore greatly accelerating the replanning. As the humanoid robot walking pattern cannot be changed instantly, a time interval t_c is required to change the planned footprints. In the implementation of the ZMP preview control, a finite time horizon of 2 steps is used to compute the CoM trajectory. Therefore, if a collision is foreseen at instant t_c, the new collision-free trajectory provided by the ARA* algorithm (in blue in Fig. 9) is deformed to keep the next two footprints unchanged (in red in Fig. 9). The robot stops if the deformed trajectory is in collision. Contrary to what Fig. 9 might suggest, 2 steps do not represent a significant distance. Since the robots are small and the feet tend to slide, 2 steps are only a few centimetres or less.

6. Results

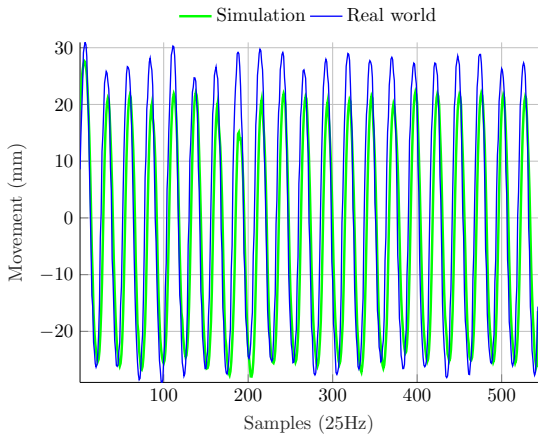
Experiments were conducted with a couple of Nao humanoid robots, manufactured by Aldebaran Robotics [40]. On top of the robots heads, we have added an Asus Xtion Pro Live consumer-level depth camera (see Fig. 1). Six strips of black tape were added to the robots in order to increase contrast in low texture areas and make their detection for the visual synchronization (see Section 4.3) easier.

The object being transported, shown in Fig. 1, is a miniature table, 600mm long by 300mm wide. The legs are cylinders, 200mm long and 40mm in diameter. The legs are too short to touch the ground, so the table is fully supported by the robots.

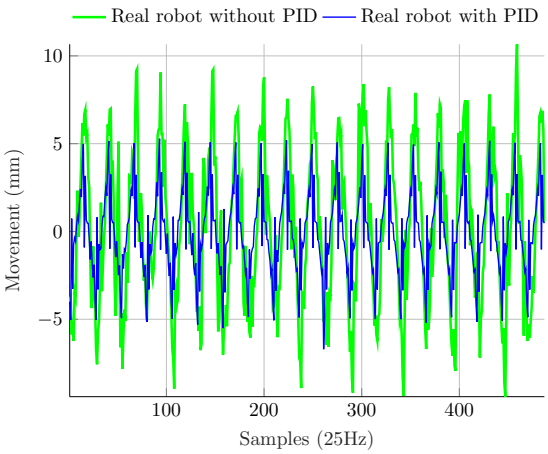
6.1. *Articulating the Arms*

Without any hand position correction, the lateral swing causes large oscillations that are transmitted to the table and the load. To prevent this problem, the hands are controlled to follow stable trajectories using the whole-body control scheme explained in Section 5.

As it can be seen in Fig. 10(a), without any correction, the average oscillation peak-to-peak movement of the hands is 47.6 mm. However, with the whole body control, the average hand distance from desired position has been reduced to 16.4 mm, reducing the hand error by 65.5%, as illustrated in Fig. 10(b). As a result, significantly less oscillations are transmitted to the table, leading to a safer and



(a) No hands correction in simulation and real world



(b) Position control and PID correction on the real robot

Fig. 10. Influence of hand corrections ($\dot{\boldsymbol{r}}_{lh}, \dot{\boldsymbol{r}}_{rh}$) on the table oscillations.

enhanced carrying ability and load stability. Moreover, to minimize the impact of the backlash and the motors time response of the Nao robot, a PID controller on the robot joint positions has been implemented. By adding the PID controller with coefficient $K_p = 1.8, K_i = 0.01$ and $K_d = -0.02$, the error has been reduced further to 78.4%, for an average peak-to-peak movement of 10.3 mm, as depicted in Fig. 10(b).

The same tests were then conducted on the real robots, and the real experiments results are consistent with simulation results. This still increases significantly the

	displacement	Observed displacement	Unmatched features (%)	Rejected by homography (%)
Moving robot perception				
Move +X	0.05 m	$0.044 \pm 1.49 \times 10^{-2}$	1.2	26.7
Move -X	−0.05 m	$-0.051 \pm 1.98 \times 10^{-2}$	7.9	24.0
Move +Y	0.1 m	$0.098 \pm 0.74 \times 10^{-2}$	2.4	18.8
Move -Y	−0.1 m	$-0.099 \pm 0.46 \times 10^{-2}$	4.9	28.2
Turn +Z rad	0.17 rad	$0.039 \pm 0.10 \times 10^{-2}$	9.8	18.9
Turn -Z rad	−0.17 rad	$-0.026 \pm 0.20 \times 10^{-2}$	13.4	26.7
Idle robot perception				
Move +X m	0.05 m	$0.041 \pm 0.65 \times 10^{-2}$	3.7	20.2
Move -X m	−0.05 m	$-0.058 \pm 0.23 \times 10^{-2}$	4.9	15.8
Move +Y m	0.1 m	$0.103 \pm 0.14 \times 10^{-2}$	12.2	14.2
Move -Y m	−0.1 m	$-0.100 \pm 0.17 \times 10^{-2}$	3.6	26.6
Turn +Z rad	0.17 rad	$0.372 \pm 1.67 \times 10^{-2}$	18.6	19.3
Turn -Z rad	−0.17 rad	$-0.324 \pm 2.46 \times 10^{-2}$	6.0	22.0

Table 2. Visual feedback static errors

load stability. Also, note that the error cannot be completely cancelled for two main reasons. Firstly, the hand trajectories are second-priority tasks only (see Section 5), which means that the robot will respect those trajectories as far as the trajectories of first priority are fully followed. Secondly, the Nao robot has only 4 DOFs in each arm.

6.2. *Visual Feedback*

The mask used for the optical flow was created by taking a picture of one Nao from the other while they are standing in position with the table in hand, then by masking the background with a uniform gray color and finally blurring some edges to reduce false corner detection, especially at the bottom of the cropped robot image. Therefore, the mask has the same size as the original image and when compared to it, the optical flow represents an absolute error value since the robot is situated at the synchronized position in the mask. By using GFTT to find interest points, with a quality level of 0.07 and a minimum distance between features of 5 pixels, we have 41 features to track, as shown in Fig. 8.

To verify the precision and robustness of the optical flow for specific movements and static configurations, the following experiments were executed by moving one robot, the observed one (the one being looked at). Meanwhile, the other (the observer, carrying the camera), was standing still. The motions made by the observed robot were linear motions in ±X and ±Y and rotations around the Z axis.

Table 2 shows a comprehensive overview of the visual feedback static errors for X, Y and Yaw. The Yaw is very inaccurate, and this is mainly because once a given object is centered in the image, small errors in the x-axis may result in large errors in the estimation of the Yaw angle. Yaw estimation is used for the moment will then be limited to an indicator for error to correct rather than an absolute error value to use as a feedback.

However, these static tests do not contain any motion blur, which is usually an

important problem with moving and oscillating humanoid robots. An example of this phenomenon is illustrated in Fig. 11. This image is taken from the experiments performed in the next section; the black lines are the features found and matching the computed homography, while the white lines are found matches that do not fit the homography model. To determine which points are valid, a Least Median of Squares (LMEDS) on the re-projection error is used. The figure illustrates the robustness to heavy blur of our technique.

Fig. 11. Demonstration of the robustness for highly blurry images

6.3. *Navigating in a Cluttered Environment*

To test the system as a whole and to validate the proposed algorithms, we conducted a series of 5 experiments. In each experiment, the robots starting and goal positions are chosen in such a way that the robots have to navigate among objects on the ground. The objects serve as obstacles to be avoided by the robots and the transported object. They are placed to form various feasible paths and force tight turns in order to take advantage of the additional degrees of freedom (the rotations of the object θ_{ob} and θ_{r_2}).

Fig. 12 shows the waypoints for each type of experiment. The start and end of each experiment are all pairs of consecutive waypoints. In Fig. 13, the obstacles are in yellow, while the red areas around them are inflation zones where the cost is higher than in free space, to prevent the robots from passing too close to obstacles (see Section 2). These zones are used as a security buffer. The centers of the robots and the object should avoid, if possible, to generate plans passing inside this zone. The cyan zone is a forbidden zone, because if the center of the object or the robots enters it, it means that an edge is in collision with an obstacle.

The ARA* planner parameter initial value $\epsilon = 3$ means that the suboptimal solution cannot be worse than 3 times the optimal solution cost. A time limit of 5 seconds was chosen and within that time, ϵ was successfully decreased to 1 on every run, which corresponds to the optimal solution. For each generated path, we measured data about the planner and the trajectories. These results are summarized in Table 3.

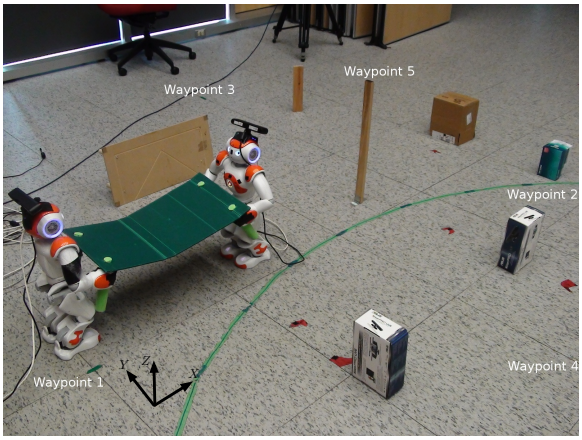


Fig. 12. Picture of the obstacles, robots and waypoints.

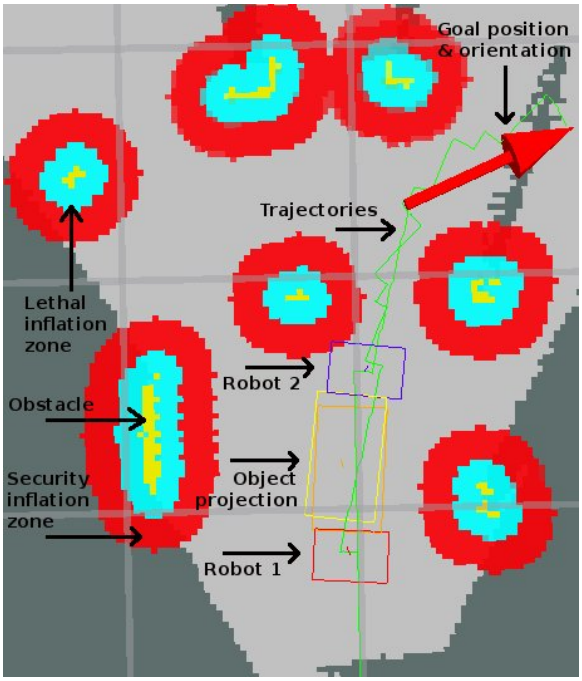


Fig. 13. Map of the starting and ending positions with obstacles, lethal and security inflation zones around them, the Nao and object footprints and goal position/orientation.

Even though in our case the trajectories are quite small, the use of sub-optimal solutions has proven to be significantly faster. Indeed, for our experimental settings, it is about 14.17 times faster and takes 17.1 times less states expansions to compute

	Mean	Std. dev.
Total execution time (s)	94.28	18.14
Trajectory length (m)	2.27	0.19
Average velocity (m/s)	0.025	0.003
Initial solution time ($\epsilon = 3$) (ms)	12	13
Optimal solution time ($\epsilon = 1$) (ms)	170	68
Initial Nodes Expansions	195.6	161.3
Total Number Of Expansions	3346.0	1674.3

Table 3. Statistics regarding the shortest possible path obtained by the motion planning algorithm

the solution for $\epsilon = 3$ as opposed to the optimal solution ($\epsilon = 1$). This could be especially useful for real life large scale distances and experiments where quick reactions and planning are necessary.

Moreover, to compare the results with those obtained by the method proposed in [24], we conducted navigation experiments while deactivating the visual feedback feature. During each experiment, the reflection error is measured at 10 Hz while the visual information is registered as fast as possible (about 14.5 Hz) when the computed homography is valid. Note that the ground-truth is a global positioning system (a Vicon system). Table 4 points out that the observation errors on the three axis have been reduced with the visual feedback.

	X (m)	Y (m)	Yaw (rad)
Reflection only[24]			
Error	5.14×10^{-2}	-2.44×10^{-2}	4.10×10^{-4}
Std	6.51×10^{-2}	3.21×10^{-2}	3.43×10^{-2}
Reflection+ Visual Feedback			
Error	7.34×10^{-3}	-1.14×10^{-3}	3.81×10^{-4}
Std	1.21×10^{-2}	1.09×10^{-2}	2.03×10^{-3}

Table 4. Comparison with the method proposed in [24]

In addition to the visual errors measured from the Naos' camera, the quality and robustness indicators previously discussed were computed and monitored. The average processing time of the visual feedback in this experiment was $68.9\pm10.5\,ms$. The results are summarized in Table 5.

It can be observed that the monitored parameters are in line with what was measured statically, even though heavy motion blur was added in the experiments. On one hand, the number of unmatched features is in the lower range of the static data. It is caused by the automatic resetting of invalid features that can more easily reattach themselves to a valid point when moving throughout the image than when

	Unmatched features (%)	Rejected points (%)	Reprojection errors (x,y) (m)
Error	5.5	25.2	$(5.49\,10^{-4}, 6.02\,10^{-4})$
Error Std	4.3	6.4	$(3.14\,10^{-3}, 3.12\,10^{-3})$

Table 5. Visual feedback errors

staying at an invalid position. On the other hand, the rejected points are in the upper range of the static equivalent. The cause is motion blur, as well as more varying background, that can both cause wrong matches.

Robustness and effectiveness of the method while in motion is demonstrated as it successfully matches and fits the majority of points even during large movements, with limited framerate and camera resolution.

The output from the SLAM library, RTAB-Map, are shown in Fig. 14. The localization information is displayed as a continuous colored line and the mapping information is represented by the rest of the incrementally built point cloud. This map could be used for any other experiment taking place at the same location, for quick localization in a known environment. It is important to note that the ground in the testing room had sufficient texture and patterns to produce reliable data for the SLAM library. When tested on a plain ground, RTAB-Map could not, however, extract enough features for visual localization using only the obstacles.

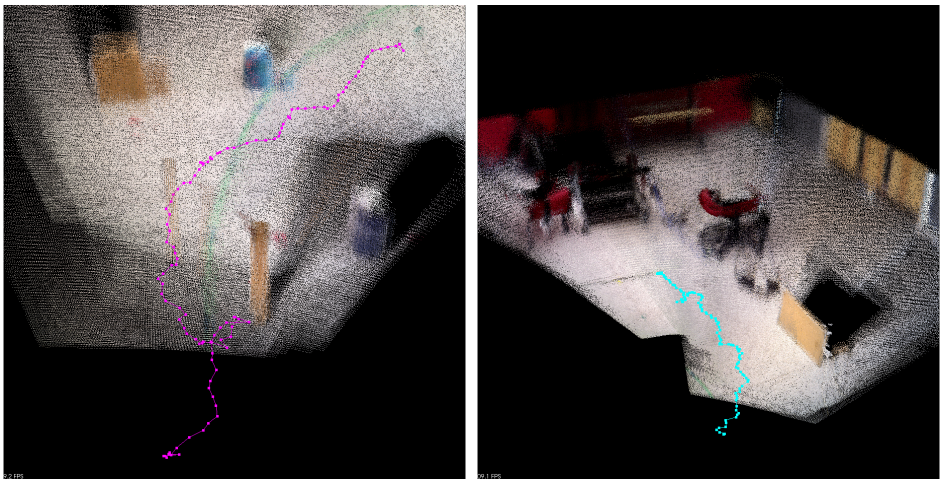


Fig. 14. Left: localization (purple line) and mapping (point cloud) close up for the backing robot. Right: Localization (cyan line) and full mapping (point cloud) for the forward robot.

With the table linking the two robots together, significant drift caused by the visual odometry imperfection can make the Nao drift when navigating as shown

in Fig. 16. This drift is mainly due to the cameras not being fixed rigidly enough on the Nao heads and slowly changing position over time. A special rigid head mount should greatly reduce this drift.

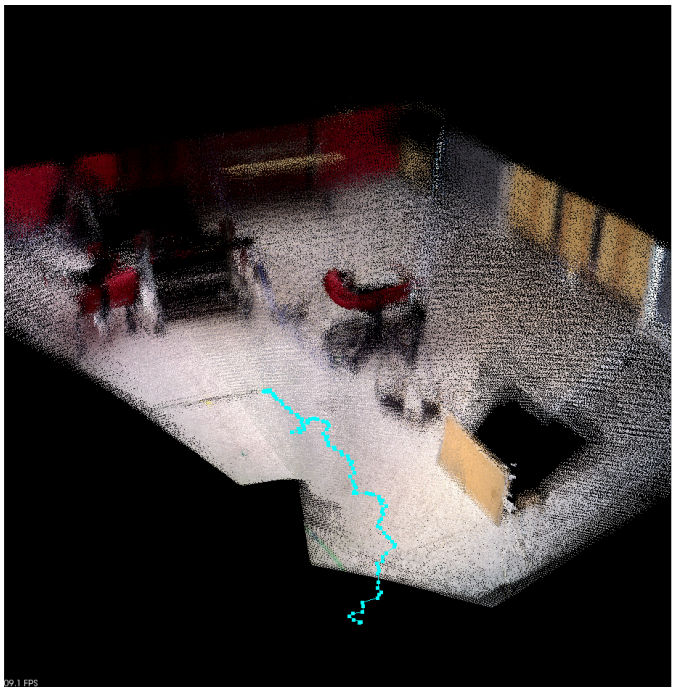


Fig. 15. Localization (cyan line) and full mapping (point cloud) for the forward robot.

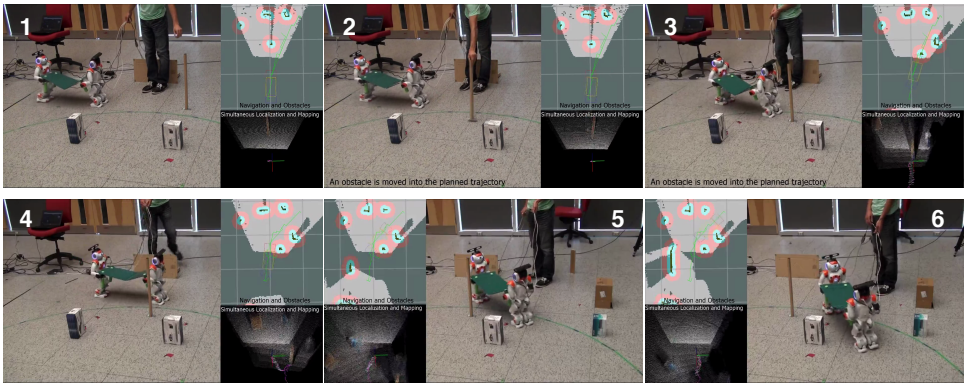


Fig. 16. Snapshots of the Naos navigating with an object in a cluttered and dynamic environment

7. Conclusion and Future Work

In this paper, we have proposed a system capable of carrying a long table with two humanoid robots while navigating in a cluttered environment; we also gave practical insights into the implementation of the proposed approach on a real humanoid robot. Our method uses a lattice-based planner designed for two robots transporting and articulating an object. This includes a reduction of the state dimensionality, a choice of adequate simple and complex primitives and a cost function computed while considering these restrictions.

When moving throughout the environment, a depth camera and a SLAM library map the obstacles in real-time and provide a visual odometry. This information is then fused with each robot odometry to provide a consistent, continuous and reliable odometry data. While mapping the environment, dynamic collisions are foreseen and trajectories modified while in motion. A finite horizon of 2 steps is however necessary before the robot can change its current trajectory. It would be possible to integrate the SLAM, the 3D occupancy grid and the loop closure algorithms provided by RTAB-Map with the memory efficient OctoMap [41] for planning and obstacle detection. However, since the project is mainly in 2D for the moment, this integration was not performed.

Moreover, by controlling the hands adequately with a whole-body control scheme coupled with a PID, we could maneuver the object in tight turns, reduce significantly the lateral swing and avoid its propagation to the object and between robots.

In future works, in order to make the robots completely autonomous, it is necessary to implement the algorithms entirely on the Nao itself to be processed by its internal CPU.

The biggest implementation challenge is the platform limited processing power in comparison with other high-end humanoid robots that have often multiple onboard computers. For example, when the visual feedback algorithms were implemented on the Naos themselves, the frame rate was reduced to 1 fps on average, which makes it too inaccurate to use.

To further improve the synchronization speed and make the system behavior more human-like, the arms should be used to absorb part of the error instead of only controlling the object for special primitives and swing reduction. In addition, the visual feedback we use for synchronization could be used to determine the drift occurring between the robots. Since they are able to see each other's real position, the visual odometry drift could be corrected to improve the long term navigation.

Another improvement would be determining the DF factor automatically, a possible strategies is to determine it as the evaluation of a cost function. This cost function could:

- Evaluate the footprint of the trajectory resulting from the application of the primitive during some specified time interval Δt at the maximal velocities;
- Sum up experimentally the energy/torques required to generate each primitive with the robot;

- Use previous human locomotion experiments [42] that have showed that the natural human walking pattern favors non-holonomic behaviors (and penalizes the sideways motion); we could integrate the functionals developed in the cited work as a penalizing weight for our primitives.

Although no strong prior information about the environment is necessary in our approach, we need to know the characteristics of the robots and of the object to carry. In the future, we will adapt the current algorithms to the much more challenging case of a collaboration between two robots who do not know each other and who have a partial knowledge of the object.

Measuring the force exchanged through the object could also be very useful in maintaining the equilibrium of the whole system. As a future work, we will address this problem using the technique that we presented in [43].

Finally, slipping avoidance and turning strategies such as the ones proposed in [44] and [45] would be investigated. Moreover, a motion planning which considers a 3D model of the environment [46] would also be studied.

Acknowledgments

We thank the editor and anonymous reviewers for their helpful and insightful comments.

This research is supported by Natural Sciences and Engineering Research Council of Canada (NSERC), and partially by a collaborative research project funded by XIVe GROUPE DE TRAVAIL QUE BEC-MEXIQUE " and a Mitacs Globalink research internship.

References

1. A. Bauer, D. Wollherr, and M. Buss. Human–robot collaboration: A survey. *International Journal of Humanoid Robotics*, 05(01):47–66, 2008.
2. J. Ota, N. Miyata, T. Arai, E. Yoshida, D. Kurabatashi, and J. Sasaki. Transferring and regrasping a large object by cooperation of multiple mobile robots. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, volume 3, pages 543–548, 1995.
3. N. Miyata, J. Ota, Y. Aiyama, J. Sasaki, and T. Arai. Cooperative transport system with regrasping car-like mobile robots. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 3, pages 1754–1761, 1997.
4. Z. Wang, M.N. Admadabadi, E. Nakano, and T. Takahashi. A multiple robot system for cooperative object transportation with various requirements on task performing. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1226–1233, 1999.
5. C.R. Kube and E. Bonabeau. Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, 30:85–101, 2000.
6. A. Yamashita, T. Arai, J. Ota, and H. Asama. Motion planning of multiple mobile robots for cooperative manipulation and transportation. *IEEE Trans. on Robotics and Automation*, 19(2):223–237, 2003.
7. ARMAR-Family. http://his.anthropomatik.kit.edu/241.php.
8. T. Asfour, K. Regenstein, P. Azad, J. Schroder, A. Bierbaum, N. Vahrenkamp, and R. Dillmann. ARMAR-III: An integrated humanoid platform for sensory-motor con-

trol. In *6th IEEE-RAS International Conference on Humanoid Robots*, pages 169–175, Dec 2006.
9. Rüdiger Dillmann, Regine Becher, and Peter Steinhaus. ARMAR II — a learning and cooperative multimodal humanoid robot system. *International Journal of Humanoid Robotics*, 01(01):143–155, 2004.
10. Y. Hirata and K. Kosuge. Distributed robot helpers handling a single object in cooperation with a human. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, volume 1, pages 458–463, 2000.
11. R. Suda and K. Kosuge. Handling of object by mobile robot helper in cooperation with a human using visual information and force information. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 2, pages 1102–1107, 2002.
12. M. Lawitzky, A. Mortl, and S. Hirche. Load sharing in human-robot cooperative manipulation. In *IEEE RO-MAN*, pages 185–191, 2010.
13. Paul Evrard and Abderrahmane Kheddar. Human-humanoid co-working in a joint table transportation. In *Proceedings of the 4th International Conference on Social Robotics*, ICSR'12, pages 357–366, Berlin, Heidelberg, 2012. Springer-Verlag.
14. K. Harada, S. Kajita, H. Saito, M. Morisawa, F. Kanehiro, K. Fujiwara, K. Kaneko, and H. Hirukawa. A humanoid robot carrying a heavy object. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1712–1717, 2005.
15. Y. Ohmura and Y. Kuniyoshi. Humanoid robot which can lift a 30kg box by whole body contact and tactile feedback. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 1136–1141, 2007.
16. E. Yoshida, C. Esteves, I. Belousov, J-P. Laumond, T. Sakaguchi, and K. Yokoi. Planning 3-d collision-free dynamic robotic motion through iterative reshaping. *IEEE Trans. on Robotics*, 24(5):1186–1198, 2008.
17. Y. Aiyama, M. Inaba, and H. Inoue. Pivoting: A new method of graspless manipulation of object by robot fingers. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, volume 1, pages 136–143, 1993.
18. E. Yoshida, P. Blazevic, V. Hugel, K. Yokoi, and K. Harada. Pivoting a large object: whole-body manipulation by a humanoid robot. *Applied Bionics and Biomechanics*, 3(3):227–235, 2006.
19. E. Yoshida, M. Poirier, J-P. Laumond, O. Kanoun, F. Lamiraux, R. Alami, and K. Yokoi. Pivoting based manipulation by a humanoid robot. *Autonomous Robots*, 28(1):77–88, 2010.
20. Yutaka Inoue, Takahiro Tohge, and Hitoshi Iba. Cooperative transportation system for humanoid robots using simulation-based learning. *Applied Soft Computing*, 7(1):115–125, 2007.
21. Meng-Hung Wu, Atsushi Konno, and Masaru Uchiyama. Cooperative object transportation by multiple humanoid robots. In *2011 IEEE/SICE Int. Symposium on System Integration (SII)*, pages 779–784, 2011.
22. Stephen G McGill and Daniel D Lee. Cooperative humanoid stretcher manipulation and locomotion. In *IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, pages 429–433, 2011.
23. Meng-Hung Wu, A. Konno, S. Ogawa, and S. Komizunai. Symmetry cooperative object transportation by multiple humanoid robots. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 3446–3451, May 2014.
24. Antoine Rioux, Claudia Esteves, Jean-Bernard Hayet, and Wael Suleiman. Cooperative SLAM-Based Object Transportation by Two Humanoid Robots in a Cluttered Environment. In *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, pages 331–337, 2015.

25. M. Likhachev, G. Gordon, and S. Thrun. ARA*: Anytime A* with provable bounds on sub-optimality. In *Advances in Neural Information Processing Systems*, volume 16, 2004.
26. C.B. Hart and S.F. Giszter. A neural basis for motor primitives in the spinal cord. *The Journal of Neuroscience*, 30(4):1322–1336, 2010.
27. O. Stasse, A.J. Davison, R. Sellaouti, and K. Yokoi. Real-time 3d slam for humanoid robot considering pattern generator information. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 348–355, 2006.
28. S. Oßwald, A. Hornung, and M. Bennewitz. Learning reliable and efficient navigation with a humanoid. In *IEEE Int. Conf. on Robotics and Automation*, pages 2375–2380, 2010.
29. D. Maier, A. Hornung, and M. Bennewitz. Real-time navigation in 3D environments based on depth camera data. In *12th IEEE-RAS Int. Conf. on Humanoid Robots (Humanoids)*, pages 692–697, 2012.
30. M. Labbe and F. Michaud. Online global loop closure detection for large-scale multi-session graph-based slam. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2661–2666, 2014.
31. M. Labbe and F. Michaud. Rtab-map project on ros.org., 2014.
32. S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa. Biped walking pattern generation by using preview control of zero-moment point. In *Proc. IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 1620–1626, 2003.
33. B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo. *Robotics: Modelling, Planning and Control*, chapter 3 (Differential Kinematics and Statics). Springer, 2009.
34. J-Y. Bouguet. Pyramidal implementation of the Lucas Kanade feature tracker. Technical report, Intel Corporation Microprocessor Research Labs, 2000.
35. J. Shi and C. Tomasi. Good features to track. In *Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600, 1994.
36. Z. Zhang. A flexible new technique for camera calibration. Technical Report MSR-TR-98-71, Microsoft Research, December 1998.
37. Antoine Rioux and Wael Suleiman. Humanoid navigation and heavy load transportation in a cluttered environment. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, pages 2180–2186, 2015.
38. F. Kanehiro, F. Lamiraux, O. Kanoun, E. Yoshida, and J.-P. Laumond. A Local Collision Avoidance Method for Non-strictly Convex Objects. In *Robotics: Science and Systems (RSS)*, 2008.
39. Adrien Escande, Nicolas Mansard, and Pierre-Brice Wieber. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. *The International Journal of Robotics Research*, 33:1006–1028, 2014.
40. D. Gouaillier, V. Hugel, P. Blazevic, C. Kilner, J. Monceaux, P. Lafourcade, B. Marnier, J. Serre, and B. Maisonnier. Mechatronic design of NAO humanoid. In *IEEE Int. Conf. on Robotics and Automation (ICRA)*, pages 769–774, 2009.
41. A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3):189–206, April 2013.
42. Gustavo Arechavaleta, Jean-Paul Laumond, Halim Hicheur, and Alain Berthoz. On the nonholonomic nature of human locomotion. *Autonomous Robots*, 25(1-2):25–35, August 2008.
43. Louis Hawley and Wael Suleiman. External Force Observer for Medium-sized Humanoid Robots. In *2016 IEEE-RAS 16th International Conference on Humanoid*

Robots, pages 366–371, 2016.
44. Jing Li, Qiang Huang, Zhangguo Yu, Xuechao Chen, Weimin Zhang, Huaxin Liu, Junyao Gao, and Yingxian Duo. Integral acceleration generation for slip avoidance in a planar humanoid robot. *IEEE/ASME Trans. on Mechatronics*, 20(6):2924–2934, Dec 2015.
45. Je Sung Yeon and Jong Hyeon Park. A fast turning method for biped robots with foot slip during single-support phase. *IEEE/ASME Trans. on Mechatronics*, 19(6):1847–1858, Dec 2014.
46. K. Harada, S. Hattori, H. Hirukawa, M. Morisawa, S. Kajita, and E. Yoshida. Two-stage time-parametrized gait planning for humanoid robots. *IEEE/ASME Trans. on Mechatronics*, 15(5):694–703, Oct 2010.

Antoine Rioux received his B.Sc. degree in Computer Engineering and M.S. degree in Electrical Engineering from the University of Sherbrooke, Canada, in 2013 and 2016, respectively. His research interests include humanoid robots, autonomous navigation, control systems and multi-robot synchronization.

Claudia Esteves is an Associate Professor at the Department of Mathematics of the University of Guanajuato, Mxico, since September 2007. She received her Ph.D. from the University of Toulouse (2007) where she worked at LAAS-CNRS on the subject of motion planning for humanoid robots and virtual characters. Her current research interests are on the motion and task planning in robotics and computer graphics, human-robot interaction and motion planning with perception constraints.

Jean-Bernard Hayet graduated from Ecole Nationale Superieure de Techniques Avancees in 1999. He got his master degree from University Paris VI and his Ph.D. degree from University of Toulouse in 2003. His doctoral dissertation tackled the problem of landmark-based navigation with computer vision in indoor environments and was supervised by Dr. M. Devy and Dr. F. Lerasle. From 2003 to 2006 he did a post-doctoral stay in visual tracking at University of Liege with Pr. Justus Piater. Since 2007 he is researcher at the Centro de Investigacion en Matematicas, in Mexico, and is focusing his research on robot vision and visual tracking.

Wael Suleiman received the Master's and Ph.D. degrees in automatic control from Paul Sabatier University, Toulouse, France, in 2004 and 2008, respectively. He is currently Associate Professor in Electrical and Computer Engineering Department, Faculty of Engineering, Université de Sherbrooke, Sherbrooke, Canada. His research interests include nonlinear system identification and control, numerical optimization, and collaborative and humanoid robots. He was a Postdoctoral Fellow of the Japan Society for the Promotion of Science (JSPS) from 2008 to 2010 at National Institute of Advanced Industrial Science and Technology (AIST), Tsukuba, Japan.