Cooperative SLAM-based object transportation by two humanoid robots in a cluttered environment

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

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

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

***Abstract*— 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 can 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 control to execute those trajectories. We present experiments conducted 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 changing the robot's main hardware, and therefore enacting the capacity of humanoid robots in real-life situations.**

I. Introduction

One of the advantages of having arms on a robot is that it can manipulate a load. This capacity can be useful for a wide range of actions, including transporting objects from one place to another. This can be particularly useful for automated construction and rescue missions situations. However, using one robot only, the maximum payload is generally low and the size of transported objects limited. One way to deal with this issue is to distribute the weight to multiple robots. In this work we deal with the problem of having two humanoid robots cooperating to handle a bulky object among obstacles.

This problem has been tackled in previous works using mainly two approaches: (1) a leader-follower control, or (2) a synchronized control. In the first approach [1] [2], one of the robots, the leader, based on its position and its surrounding, computes the plan for the system or is directly guided by a human operator. The second robot follows the leader. This technique is easy to implement, but does not allow closed-loop cooperation easily, because, as the follower only responds to the leader movement when it has already started, a significant time delay is introduced.

Most human-robot cooperative system use this approach [3] [4] since these problems are alleviate by adding a human into the loop, which instinctively correct both micro and macro errors in the system.

In the second approach [5] [6], an external centralized controller computes the motion for all the robots simultaneously, based on the information of the environment provided by the robots. As a result, the synchronized motions can start and stop together. However, both synchronization system rely on "step synchronization" instead of position synchronization. While easier to synchronized, this process slows down the system significantly and increased lateral instability as the entire system swing from side to side in harmony. Furthermore, the transported object is held rigidly and cannot be articulated to improve the footprint and motions of the system. In this work, which belongs to the latter approach, a framework for synchronized cooperative autonomous humanoid robots navigating in a cluttered environment while manipulating an object in closed-loop is presented.

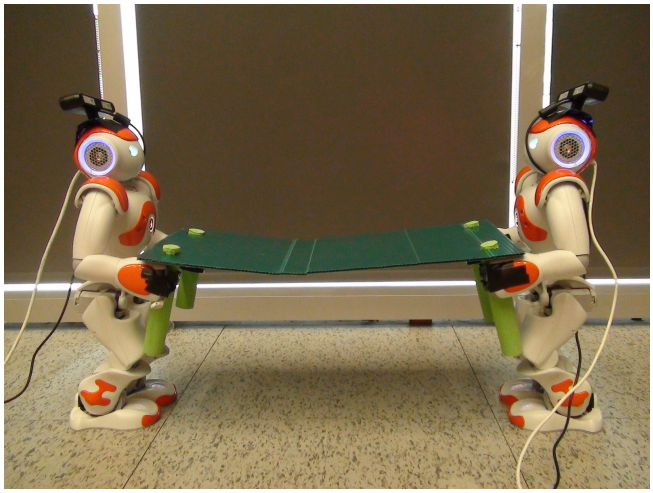


Fig. 1. The Nao robots holding an object together

Our main contributions are: (1) a low dimensional multi-robot motion planning algorithm to find an obstacle-free trajectory using a map of the environment autonomously constructed by the robots, (2) a continuous and consistent odometry system integrating the robots visual data and actuators information, (3) a synchronization strategy that uses the projection of the robots, and (4) an efficient real-time whole-body control scheme that generates the motions of the closed-loop robot-object-robot system. The remainder of the paper is organized as follows: Section II presents the proposed planning algorithm. In Section III, a brief description of how the map of the environment is constructed is given. Section IV details the proposed synchronization approach and Section V describes the control scheme to execute the planned trajectories. Finally, in Section VI, results of simulation and real world experiments are presented and discussed.

A. Rioux and W. Suleiman – Electrical and Computer Engineering Department, Faculty of Engineering, University of Sherbrooke, Canada; C. Esteves – Department of Mathematics. Universidad de Guanajuato, México; J-B. Hayet – Centro de Investigación en Matemáticas (CIMAT), Guanajuato, México. {antoine.rioux, wael.suleiman}@usherbrooke.ca, {cesteves, jbhayet}@cimat.mx

II. 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 [7], motivated by the use of motion primitives ensuring feasible robot-object-robot configurations and transitions. The environment is modelled by a 2D grid cost-map that discriminates obstacles from free space at a fixed threshold and allows obstacles inflation to increase the security margin.

A. State representation

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

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

where x_{ri}, y_{ri} and θ_{ri} $(i = 1, 2)$ are the positions and orientation of the i-th robot, and x_{ob}, y_{ob} and θ_{ob} are those of the object. 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 shown in Fig. 2. The closed-loop grasping of the robot on the table is shown in Fig. 1. The pivot points positions maximize the rotation range within the robot workspace, resulting in a smaller 5 dimensions state space $\mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{S}^1 \times \mathbb{S}^1$:

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

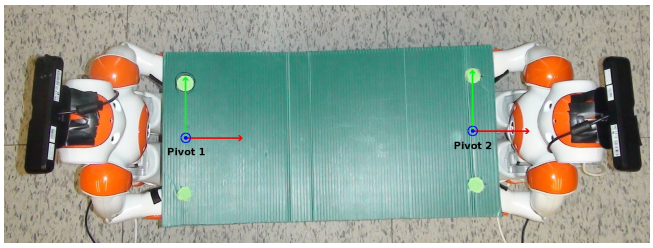


Fig. 2. Top view: Pivots position

Even though the above simplification removes the ability of the object to translate on the plane, 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.

In a lattice-based graph planner, transitions between nodes are triggered by actions chosen within a finite fixed-set of motion primitives. An important feature of the lattice representation is that all connections are feasible paths. Therefore, it 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 graph search, including Von Neumann or Moore neighborhood.

The set of motion primitives used for this problem is showed in Fig. 4. It includes (a) forward, backward, sideway motion and every diagonal motion, (b) rotations around each robot and the object center and special movements such as

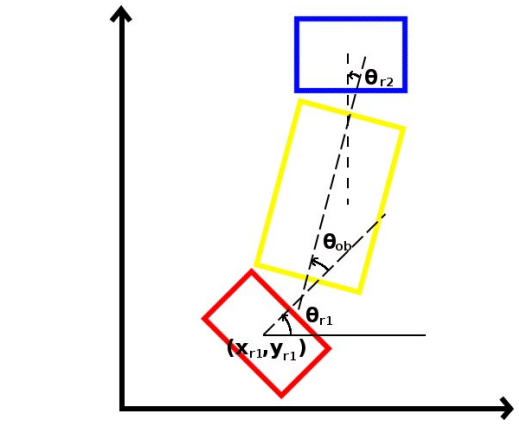
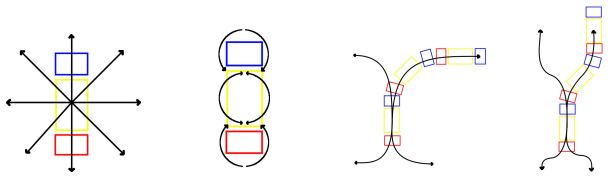


Fig. 3. Simplified state representation.



(a) Linear (b) Rotation (c) C-turn (d) S-turn

Fig. 4. Set of possible motion primitives.

(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 over homogeneous sampling of the system DOFs. The joints of the system that allow these configurations are the aforementioned pivot points.

B. Path Cost Function

The cost of a transition from state s to s' is based on the time to execute that transition and is computed as follows:

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

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, $\dot{\boldsymbol{r}}^+$ is the robot maximal linear velocity. This ratio gives us the approximate time to execute the primitive. Since the state representation does not contain any information about the position of the second robot, we must determine Δx_{r2} and Δy_{r2} with the first robot position. To do so, we compute the projection of the object by rotating the first robot by θ_{r1}, given a hand rotation of θ_{ob}. Then, we project the second robot around the object given a hand rotation or θ_{r2}.

The difficulty factor DF is a special value set by the system expert in order to prioritize or penalize certain motions. For example, turning in place then going forward takes longer to execute than moving in diagonal. For relatively long

distance, however, moving in a straight line has a smaller trajectory footprint, is more natural looking, minimizes walking oscillation and causes less drift and slippage than moving in diagonal. For those reasons, all the diagonal primitives have higher DF than turning in place and moving forward. The difference in cost will still, however, favor a diagonal movement where turning in place is not worth it, i.e. for short diagonal movements.

C. 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 A* is optimal when it finds a solution, that solution does not always exist or cannot be found within a reasonable time. The Anytime Repairing A* (ARA*) focuses on delivering a suboptimal solution as fast as possible; this 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 after replanning and do not need to be recomputed. The cost function takes the form of:

$$f(s,s') = g(s,s') * max(Cost_{cells}(s,s')) + \epsilon h, \epsilon >= 1 \quad (4)$$

where $g(s,s')$ is the path cost of equation (3), h is the heuristic that uses a 2D grid containing all the Dijkstra distance costs from the goal to the start states and

$$Cost_{cells}(s,s') = \begin{cases} 1 & \text{free space} \\ 2 \text{ to } 99 & \text{inflation} \\ \infty & \text{obstacles} \end{cases} \quad (5)$$

includes the cost of cells between s and s'. The search is biased towards states closer to goal and returns a solution that is, at worst, ϵ times the cost of the optimal solution.

The inflation 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 danger of collisions. It can be set as a decreasing gradient from the obstacles or a fixed value in the range above.

III. Simultaneous Localization and Mapping

To navigate in a cluttered environment, robust and precise sensing is primordial to determine the position of obstacles, detect collisions and to plan valid paths. Also, odometry drift must be constantly corrected by an accurate localization to ensure that planned paths are closely followed. However, the Nao robot has only two cameras in its head for sensing and using them for localization has proven to be a very difficult task [8], because of the robot sway motion and the low resolution pictures. Furthermore, in our case, the field of view of the Nao is greatly obstructed by the object being transported and by the other robot.

A. Real-Time Appearance-Based Mapping (RTAB-Map)

We chose to add a depth camera on the top of each Nao's head for mapping [9], based on RTAB-Map [10], [11]. RTAB-Map provides a robust odometry system based on visual information. It can also create 3D maps of the environment as well as constructing a 2D occupancy grid map by projecting the obstacles on the ground plane.

B. Odometry fusion

A problem that occurs when using the visual odometry produced by RTAB-Map is that it may lose track of the position for multiple reasons, such as lack of detected 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. For this reason, a fusion of the visual odometry, the robot's internal odometry and the error between those reference frames is used to improved the overall odometry.

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

$$\begin{aligned} T_a^o &= T_a^v * T_v^o * T_r^{o-1} * T_r^o \\ &= T_a^v * T_v^r * T_r^o, \end{aligned} \quad (6)$$

where T_r^v 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 occurs at $t = t_{lost}$,

$$T_a^o(t) = \begin{cases} T_a^v * T_v^r(t) * T_r^o(t) & \text{if } T_v^o \text{ exists,} \\ T_a^v * T_v^r(t_{lost}) * T_r^o(t) & \text{otherwise.} \end{cases} \quad (7)$$

This approach consistently provides smooth odometry. 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.

IV. Synchronization

A. Object Stability and Hand Stabilization

At low speed, a humanoid robot CoM moves horizontally from one support foot to the other in order to stay in balance. This lateral 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, it is possible to reduce this effect so as to improve the closed-loop stability. On the one hand, if both robots swing at the same time, synchronization is maintained without effort, 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 this movement will be transmitted to the other robot and cause instability or fall.

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 at the starting position of the robot and corresponds to the transformation between feet and hands. Those transformations will be the input of the whole-body control scheme described in Section V.

B. Synchronized trajectories

Even if both robots are independent and receive their own trajectory to follow, these trajectories are lined up in such a way that they are at a constant distance from each other. However, real robots being imperfect, do not necessarily move at the exact same speed given the same command. For this reason, the trajectory is segmented into waypoints, each waypoint being a state of the graph plan. To progress to the next waypoint, both robots have to agree that they are close enough to their current waypoint in term of position and orientation. If only one robot is near its waypoint, it will slow down while converging to it, until the other robot agrees that they can proceed to the next waypoint.

C. Synchronized projections

Now that the whole system is more stable with regards 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 projection of each robot with respect to the other is computed by using their 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 and the object frames, the robot and its hands and the object center. The function $midpoint$ computes the midpoint of two points.

With these transformations, we find the projection 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 projected robot i position in the world frame and $T_{r_i{}^c}^{o}$ is the odometry data from the other robot i^c. 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 projected position to the actual position of each robot in order to find the projection synchronization error e_{p_i}

$$e_{p_i} = T_{p_i}^{o} * T_{r_i}^{o\,-1}. \tag{10}$$

D. Synchronization as Kinematics tasks

Our objective is to express the synchronization between the two robots as kinematics tasks that can be solved using a whole-body control scheme:

- The hands synchronization can be expressed as a kinematics task on the hands positions of each robot.
- The synchronized projection can be expressed as a kinematics task on the chest frame of each robot. More precisely, this kinematic task is on the horizontal positions (x and y) and the orientation around the vertical axis (z) of the chest frame.

V. Control

Once a collision-free trajectory is found by the ARA* algorithm, a set of footprints are defined along the trajectory as shown in Fig. 5. The second step is to define a Zero Moment Point (ZMP) trajectory. A trajectory of the Center of Mass (CoM) of the robot is then obtained using the preview control algorithm proposed in [12]. 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 as well as to follow the object orientation.

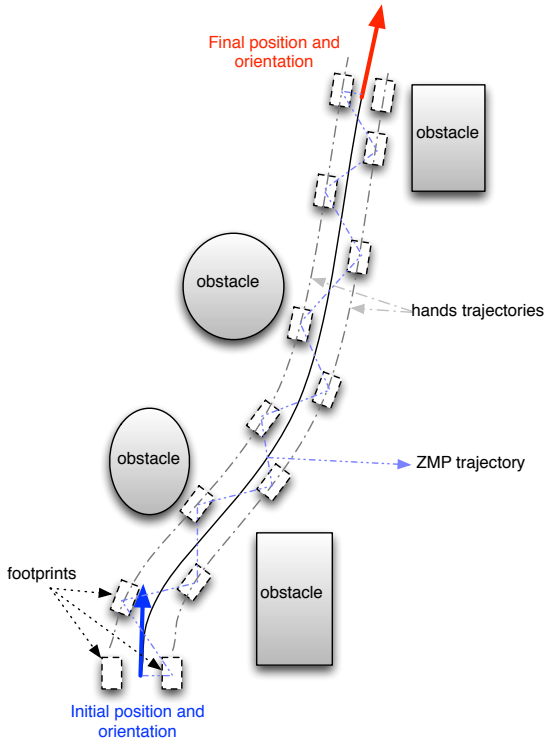


Fig. 5. Overview of the motion planning procedure

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

$$\min_{\dot{\boldsymbol{q}}} \dot{\boldsymbol{q}}^T \boldsymbol{Q} \dot{\boldsymbol{q}}$$

subject to

$$\begin{aligned}
&\text{First priority} && \begin{cases} \boldsymbol{J}_c\,\dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_c \\ \boldsymbol{J}_{lf}\,\dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{lf} \\ \boldsymbol{J}_{rf}\,\dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{rf} \end{cases} \\
&\text{Second priority} && \begin{cases} \boldsymbol{J}_{lh}\,\dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{lh} \\ \boldsymbol{J}_{rh}\,\dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{rh} \\ \boldsymbol{J}_{ch}\,\dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{ch} \end{cases} \\
&\text{Joint velocity limits} && \dot{\boldsymbol{q}}^- \leq \dot{\boldsymbol{q}} \leq \dot{\boldsymbol{q}}^+
\end{aligned} \tag{11}$$

where $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is the joint velocity vector, $\boldsymbol{Q}$ is a positive semi-definite matrix, $\boldsymbol{J}_c \in \mathbb{R}^{3\times n}$, $\boldsymbol{J}_{lf} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{rf} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{lh} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{rh} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{ch} \in \mathbb{R}^{3\times n}$ are the Jacobian matrices of CoM, left foot, right foot, left hand, right hand and chest, respectively. $\dot{\boldsymbol{r}}_c$, $\dot{\boldsymbol{r}}_{lf}$, $\dot{\boldsymbol{r}}_{rf}$, $\dot{\boldsymbol{r}}_{lh}$, $\dot{\boldsymbol{r}}_{rh}$, $\dot{\boldsymbol{r}}_{ch}$ are the linear and angular velocity of CoM, left foot, right foot, left hand, right hand and chest, respectively. Since we are only interested in the horizontal velocity and angular velocity around the vertical axis for the chest frame, $\dot{\boldsymbol{r}}_{ch} \in \mathbb{R}^3$, and $\dot{\boldsymbol{q}}^-$ and $\dot{\boldsymbol{q}}^+$ are the joint velocity limits.

The optimization problem (11) can be transformed into a standard Quadratic Programming (QP) problem [13], which can be solved in real-time by using an appropriate QP solver.

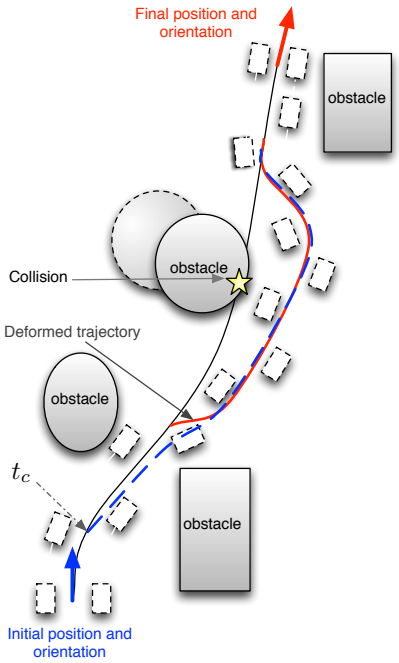


Fig. 6. Replanning in case of collision detection: t_c is the instant at which a collision is foreseen, the new collision-free trajectory is in dashed-blue line, the deformed trajectory is in red line.

A. Dynamic collision avoidance and Replanning

It is important to check frequently for collision in case an obstacle has moved from its original position, the robots drift away from their planned trajectory or the synchronization process of the robots requires a replanning of the whole 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 is necessary and the trajectory is deformed as shown, for the purpose of clarity for a single robot, in Fig. 6. 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 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 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 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 is deformed to keep the next two footprints unchanged as shown in Fig. 6. The robot will however stop if the deformed trajectory is in collision. Contrary to what Fig. 6 might suggest, 2 steps do not represent a significant distance. Since the robots are very small and the feet tend to slide, 2 steps are only a few centimetres or less.

VI. Results

Experiments were conducted on Nao humanoid robots (Fig. 1), manufactured by Aldebaran Robotics [14]. Their dimensions are 573mm of height, 311mm of width and 275mm of depth for a total weight of 5.2kg. The two arms as well as both legs have 5 DOF each, while the head has 2 DOF and the pelvis and hands have 1 DOF each. An IMU provides odometry data and 36 magnetic rotary encoders give joint angle information with a precision of 0.1°. On top of their heads, we have added an Asus Xtion Pro Live consumer-level depth camera (see Fig. 1).

A. Articulating the arms

Without any correction, the average oscillation peak-to-peak position 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 a result, significantly less oscillations are transmitted to the table, leading to a safer and enhanced carrying ability and load stability. However, the error cannot be completely cancelled, this is mainly because: I) the hands trajectories are second priority tasks, II) Nao has only 4 DOFs in each arm.

B. Navigating in a cluttered environment

To test the system as a whole and to validate the proposed algorithms, we conducted two 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. Each of them serves as an obstacle to be avoided by the robots and the 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 rotation of the object θ_{ob} and θ_{r_2}). The two series include the same experiments, except for one series that has the active projection correction and the trajectory synchronization while the other has only the latter.

Fig. 7 shows the start and end positions for each type of experiment. In this figure, 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. These zones are used as a security buffer and the center of the robots and the object should avoid, if possible, planning to pass inside it. 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 the total time to execute the trajectory, the trajectory length, the initial solution and optimal solution times. These results are summarized in Table I.

	No Sync	With Sync
Total time (s)	83.38	81.82
Total time Std (s)	9.48	9.00
Trajectory length (m)	2.36	2.28
Trajectory length Std (m)	0.30	0.22
Average velocity (m/s)	0.0283	0.0278
Initial solution time ($\epsilon = 3$) (s)	0.026	0.028
Initial solution time Std (s)	0.036	0.04
Optimal solution time ($\epsilon = 1$) (s)	0.352	0.322
Optimal solution time Std (s)	0.305	0.361

TABLE I

STATISTICS ABOUT THE EXPERIMENTS

We observe that the average speed is nearly the same, which is surprising because it was expected to be significantly slower with the projection synchronization. Indeed, for the robots to properly stay synchronized, they try to match their speed and relative position while following their respective trajectory. This means that the fastest robot slows down to accommodate the other while the slowest tries to speed up. Since the difference in average velocity is only about 1.8%, it could also be explained by other factors, such as different battery level and motors temperature.

The slowest robot was most likely already slowing down the fastest one by not agreeing to pass the waypoints, forcing a slow down. The projection synchronization forced the slow one to move through these waypoints faster, causing the slow down on the fastest to be mostly cancelled by these faster waypoint transitions.

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

In Table II, the errors in X, Y and Yaw for both robots were measured at 10 Hz during each experiments.

	Robot 1			Robot 2		
	X	Y	Yaw	X	Y	Yaw
No proj (m)	0.032	0.024	-0.009	0.033	0.0175	0.009
No proj Std (m)	0.0025	0.007	0.013	0.002	0.011	0.013
Projection (m)	0.010	0.003	0.006	0.011	0.007	-0.006
Projection Std (m)	0.001	0.004	0.0125	0.002	0.007	0.0125
Error reduction (%)	68.75	87.50	33.33	66.67	60.00	33.33

TABLE II

ERRORS STATISTICS

The error is significantly reduced on every axes for both robots. More importantly, the errors in Y and Yaw converge to zero. However, we still have some error in the X axis, because it is the main direction of movement and the system wants to follow the planned trajectories as a main priority. The focus here is to reduce the errors, but not necessarily cancel them completely if it means that it will prevent normal motion or highly impair it. As shown previously, the average velocity barely changes, while the error reduction has been greatly improved.

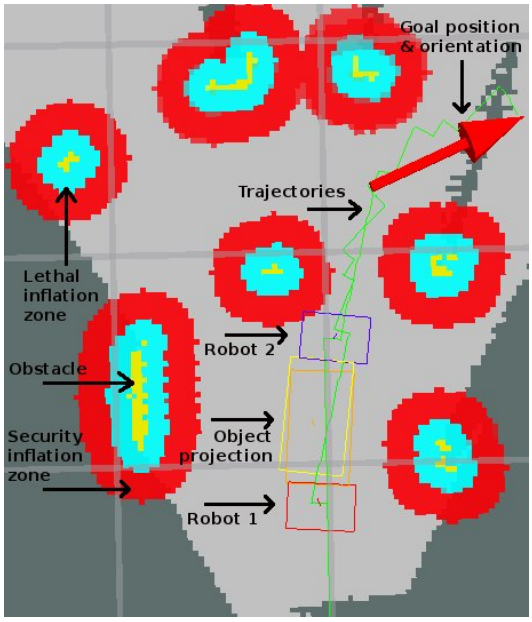


Fig. 7. Map of the starting and ending positions with obstacles, lethal and security inflation zones around them, the Nao and object footprints and goal position/orientation.

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 with the obstacles only.

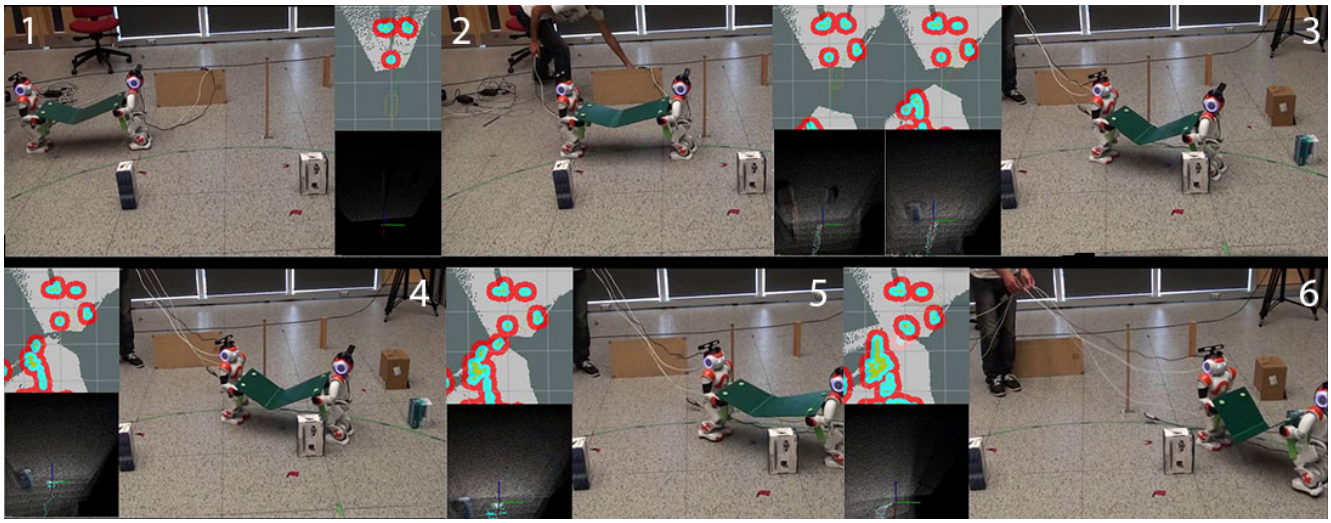


Fig. 8. Snapshots of the Naos navigating with an object

With the table linking the too robots together, significant drift cause by the visual odometry imperfection can make the Nao drift when navigating as shown in Fig. 8.

VII. 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. 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. Moreover, by controlling the hands adequately by using a whole-body control scheme, we were able to articulate the object in tight turns and to significantly reduce the lateral swing from propagating to the object and between robots.

In future works, in order to make the system more reactive and human-like, the arms should be used to absorb a part of the error instead of having to quickly move in fear of being unbalanced. In addition. the cameras in front of the Nao could be used to look at the other robot's relative position, instead of relying on hand position. As it is now, each robot can never really verify its relative position with respect to the other robot, and, as a result, it cannot detect drift errors in the visual odometry.

Acknowledgment

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 QUÉBEC-MEXIQUE"* and a Mitacs Globalink research internship.

References

[1] 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.

[2] Meng-Hung Wu, Atsushi Konno, and Masaru Uchiyama. Cooperative object transportation by multiple humanoid robots. In *System Integration (SII), 2011 IEEE/SICE International Symposium on*, pages 779–784, 2011.

[3] Kazuhiko Yokoyama, Hroyulu Handa, Takakatsu Isozumi, Yutaro Fukase, Kenji Kaneko, Fumio Kanehiro, Yoshihim Kawai, Fumiaki Tomita, and Hirohisa Hirukawa. Cooperative works by a human and a humanoid robot. In *Conference on Robotics and Automation, 2003. Proceedings. ICRA'03. IEEE International*, volume 3, pages 2985–2991, 2003.

[4] Antoine Bussy, Pierre Gergondet, Abderrahmane Kheddar, François Keith, and André Crosnier. Proactive behavior of a humanoid robot in a haptic transportation task with a human partner. In *RO-MAN, 2012 IEEE*, pages 962–967, 2012.

[5] Stephen G McGill and Daniel D Lee. Cooperative humanoid stretcher manipulation and locomotion. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 429–433, 2011.

[6] Tomoaki Yoshikai, Takahiro Akimoto, Kaoru Kobayashi, Jumpei Tsuji, Hiroaki Yaguchi, and Masayuki Inaba. Achievement of 'mikoshi'with multiple humanoid robots as coordinated navigation problem based on real-time 3d space recognition in a dynamic environment. In *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on*, pages 859–866, 2012.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] M. Labbe and F. Michaud. Rtab-map project on ros.org., 2014.

[12] 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.

[13] Antoine Rioux and Wael Suleiman. Humanoid navigation and heavy load transportation in a cluttered environment. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2015.

[14] 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.