Autonomous SLAM based humanoid navigation in a cluttered environment while transporting a heavy load

Antoine Rioux, Wael Suleiman

Autonomous SLAM Based Humanoid Navigation in a Cluttered Environment while Transporting a Heavy Load

Antoine Rioux, Wael Suleiman

A. Rioux and W. Suleiman are with Electrical and Computer Engineering Department, Faculty of Engineering, Université de Sherbrooke, Sherbrooke, Canada

Abstract

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 objects. This is especially useful when transporting loads with variable weights and shapes without having to change the robot hardware. Inspired by the wide use of makeshift carts by humans, we tackle, in this work, the problem of a humanoid robot navigating in a cluttered environment while displacing a heavy load that lies on a cart-like object. 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 these trajectories. Our contributions are as follows: (1) a whole-body control scheme that makes the humanoid use its hands and arms to control the motions of the cart-load system (e.g. tight turns) (2) a sensorless approach to automatically select the appropriate primitive set according to the load weight (3) a motion planning algorithm to find an obstacle-free trajectory using the appropriate primitive set and the constructed map of the environment as input (4) an efficient filtering technique to remove the cart from the field of view of the robot while improving the general performances of the SLAM algorithms and (5) a continuous and consistent odometry data formed by fusing the visual and the robot odometry information. We present experiments conducted on a real Nao robot, equipped with an RGB-D sensor mounted on its head, pushing a cart with different loads. Our experiments show that the payload can be significantly increased without changing the robot's main hardware, and therefore enacting the capacity of humanoid robots in real-life situations.[1]

Keywords: Humanoid robot; Localization and mapping; Navigation; Whole-body control; Motion planning

Email address: Wael.Suleiman@USherbrooke.ca (Wael Suleiman)

[1] A preliminary version of this paper has been presented at the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2015) and published in [30].

1. Introduction

One of the advantages of having arms on a robot is that it can carry a load. This capacity can be useful for a wide range of actions, including transporting objects from one place to another. However, the maximum payload is generally pretty low and generates a lot of instability when held at the arm's length. While it is possible to increase the strength of the motors in the legs and arms, it is not the best solution since a motor's power is proportional to its size, weight and price. Instead of putting the entire load directly on the robot, a cart-like object can be used to help support the weight. The structure can then be pushed by the robot and moved around more easily without having to modify the robot's hardware to fit the load.

Many studies have been done on robots moving objects to a specific goal. Most of them though are executed with multiple wheeled robots that position themselves around the object to push it in the desired direction [19, 25, 29, 36, 37]. In these researches, the manipulator comes in contact with the object at only one point, which barely allows full control of the object while moving and manipulating it. Holonomic wheeled robots are less complex to control than humanoid robots and are mainly used here to slide box-like object on the ground. Robots with humanoid arms have better control on the structure of an object and are therefore more suitable to control various shaped objects.

More advanced and specialized models of wheeled robots can possess humanoid torsos and arms which give them the same capacity to keep a high level of control when holding or transporting objects [12, 14, 22, 35]. However, most environments, made by and for humans, are more suitable for humanoid robots. To this end, the subject of bulky or heavy objects transport with a humanoid robot has received attention in the past years. To achieve this task, many different techniques have been developed. For instance, lifting the load from the ground [8, 27, 39] or using a part of the transported object as a pivot to move it [1, 38, 40].

Other works have explored the usage of humanoid robots that push heavy objects that are placed on a cart while keeping a firm grip on the handles. In theory, two arms are enough to fully constrain and control all the degrees of freedom of a cart. It was demonstrated that Honda's ASIMO is capable of moving a large cart in rooms and hallways [32] and a HRP-2 [18] is able to push a person on a wheelchair [26]. However, in both cases, the cart is mostly controlled as a Dubins car [2] instead of taking advantage of the full possibility of the humanoid robot's holonomic movement. Also, the arms serve only as a mean of attaching the cart to the robot and are not taken into consideration to control the cart further.

In [34], a planning method for humanoids to navigate among movable obstacles has been proposed. The main purpose of that method is to find a path from a starting to a goal points in a complex environment where the robot can easily

move objects to create a clear path, if it exists. Our objective is however different, as we are interested in not only navigating in a cluttered environment, but also transporting a heavy load that is significantly bigger than the humanoid payload.

In the work of [31], a PR2 robot possessing a wheeled holonomic base and two 7 DoF (Degree of Freedom) arms is used to push a small cart and control its orientation. However, despite having humanoid arms to orient the cart, since the PR2 has wheels instead of legs, no lateral swing is transmitted to the transported object causing oscillations and instability, which is a problem with humanoid robots. Furthermore, the cart is very small and light weight with respect to the PR2, in contrast to our proportionally big cart that is able to carry a load heavier than the robot itself.

The main contribution of our work is providing a complete framework for autonomous humanoid robot navigation in a cluttered environment while manipulating a cart-like object that carries a heavy load. In addition, an efficient swing reduction control algorithm and an automated sensorless primitive set selection based on the load weight are explained. Finally, an optimized algorithm to filter out the cart and an improved odometry data are proposed.

To address the problem of navigating a cart-like object supporting a load with a humanoid robot in a cluttered environment, the following sub-problems should be solved: (I) planning, (II) controlling the cart-like object and (III) sensing the environment. 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, which consider both the robot and the cart footprint in order to plan a safe trajectory between obstacles. Section 3 describes how to find the humanoid robot's footprints, and then compute the humanoid's feet and hands trajectories in order to minimize the swing effect and follow the cart trajectory using a task priority whole-body control scheme. In Section 4, real-time information from a consumer level depth camera is used to perform simultaneous localization and mapping (SLAM) of the surrounding obstacles in the cluttered environment. Finally, in Section 5, results of simulation and real world experiments are presented and analyzed.

2. Planning a valid path

To be able to navigate through a cluttered environment, a path provided by a motion planning algorithm is essential. Among the different possibilities, we chose a lattice-based graph planning with an ARA* search [23]. This choice is mainly motivated by the use of motion primitives that assures feasible robot-cart configurations and transitions. The environment is modeled by a 2D grid costmap that discriminates obstacles from free space at a fixed threshold and allows obstacles inflations to increase the security margin.

2.1. State representation

Each node of the search graph needs a complete state representation of the robot-cart to properly operate. To achieve this, it is possible to model the state

in $\mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{R}^2 \times \mathbb{S}^1$:

$$s = (x_r, y_r, \theta_r, x_{ca}, y_{ca}, \theta_{ca}) \tag{1}$$

Where x_r, y_r and θ_r are the positions and orientation of the robot, x_{ca}, y_{ca} and θ_{ca} are those of the cart. The problem can be, however, simplified by setting a pivot point positioned in the middle of both hands to reduce the dimensionality of the search space, this simplification is valid in our case because: I)- the working space of our robot's arms is too small to fully take advantage of both rotation and translation, II)- we consider that the robot is always grasping the cart handles during the execution of the trajectory. The closed loop grasping of the robot on the table is shown in Fig. 8. The chosen position of the pivot point maximizes the rotation range within the robot's workspace (see **pivot point 1** in Fig. 11), resulting in a 4 dimensions state space $\mathbb{R}^2 \times \mathbb{S}^1 \times \mathbb{S}^1$:

$$s = (x_r, y_r, \theta_r, \theta_{ca}) \tag{2}$$

Even-though the above simplification removes the ability of the cart to translate on the plane, the robot retains enough manipulability to minimize the cart footprint on tight turns. The simplified state representation of equation (2) is shown in Fig. 1.

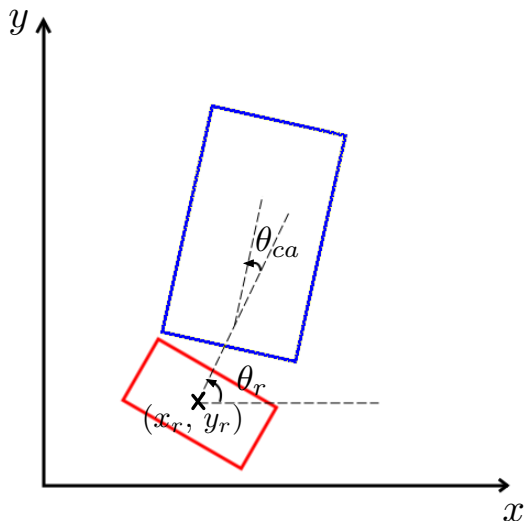


Figure 1: Simplified state representation

2.2. Transition model

In a lattice-based graph planner, the transition between the nodes is a discrete action chosen within a fixed-set of possible actions called motion primitives. Motion primitives allow the decomposition of complex motion generation

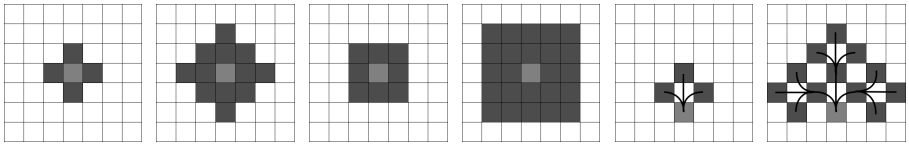
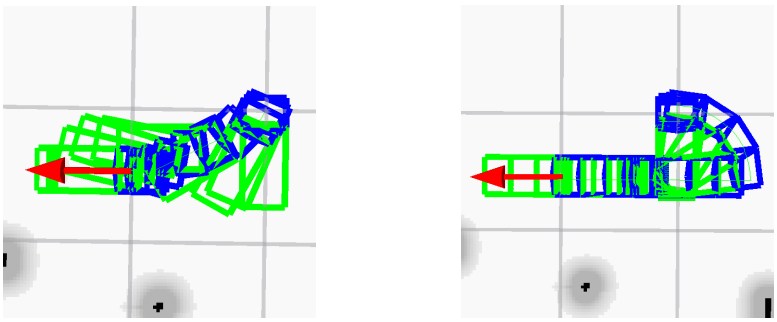


Figure 2: Different forms of graph search methods. From left to right: Von Neumann neighborhood 1st & 2nd expand, Moore neighborhood 1st & 2nd expand, example of primitives set 1st & 2nd expand



(a) omnidirectional primitives set

(b) heavy load primitives set

Figure 3: A right turn executed by the robot (blue) pushing the cart (green) to the goal (red arrow) with both sets of primitives.

in robotics and it is very likely that such arrangement of atomic motions are also used by humans and animals [6, 9]. An important feature of the lattice representation is that each of those connections is a feasible path, in contrast to other commonly used forms of graph search, including Von Neumann neighborhood or Moore neighborhood. An example of the first two expands of these graph search methods is shown at Fig. 2. This makes it really suitable for highly constrained systems, such as a robot moving a cart.

Because the cart can carry different loads, multiple sets of primitives are needed depending on the load weight. Without load, the robot is holonomic and can move in any direction. A subset of movements composed of forward, backward, diagonal, rotate in place and turn while moving forward is used to reduce planning time while focusing on forward movements. With a heavy load though, moving sideways and rotating in place becomes really difficult because of the increased friction. For this reason, when the weight becomes too important, the robot's rotation occurs around a pivot point situated between the two table legs touching the ground (see **pivot point 2** in Fig. 11). Thus, changing the feasible primitives is necessary for the lattice representation to remain coherent. An example of right turns for the omnidirectional and the heavy load sets of primitives are presented in Fig. 3.

2.3. Automatic selection of primitive sets

In order to determine which set of primitives is the most appropriate to be used, an estimation of the load weight is necessary. The easiest solution to automatically determine this value is by laying multiple pressure sensors on the table surface. However, this solution is impractical and requires adding external hardware to the robot. Instead, path following trials are performed by the robot upon startup. Indeed, since the friction creates a pivot point at high weight, the proper motions of the robot are hindered. For instance, if a command to turn in place, walk sideways or turn the table is sent, it won't be possible for the robot to correctly execute it. As a result, the error between the internally computed motion of the robot and the visually perceived one would increase proportionally to the load weight.

Depending on the robot, table and load, multiple basic motion candidates can be used and need to be evaluated.

- Walking forward in the sagittal plane (X direction), because greater loads should cause proportionally more feet slippage.
- Walking in the lateral direction (Y direction), because greater loads should cause the robot to proportionally turn around the wheels pivot instead of walking straight.
- Turning in place, because greater loads should proportionally reduce the angular rotation.

Then, in order to be able to use this movement error, it is necessary to have at least one threshold to separate the categories. For the time being, there are only two categories; the first with zero to low weight that does not impair too much the movement and the second with a heavy load that highly restricts the possible motions. Therefore, we have the following two hypotheses:

1. Hypothesis H_0: the carried load weight is low and the most appropriate primitive set is omnidirectional primitives.
2. Hypothesis H_1: the carried load weight is heavy and the most appropriate primitive set is heavy load primitives.

Let us suppose that the probability distribution functions of H_0 and H_1 are Pr_{H_0} and Pr_{H_1} respectively. A threshold x, that minimizes the probability of false detection of the correct hypothesis, can be defined as the point at which Pr_{H_0} and Pr_{H_1} are equal

$$Pr_{H_0}(\mathcal{X} = x) = Pr_{H_1}(\mathcal{X} = x) \tag{3}$$

Where $\mathcal{X}$ is a random variable. An example of the application of this method is given in Fig. 4.

Using this method, it is easy to find N-1 threshold in the case of N primitive sets. However, since every distribution gets closer to each other and start to overlap as N increases, the probability of false detection also increases.

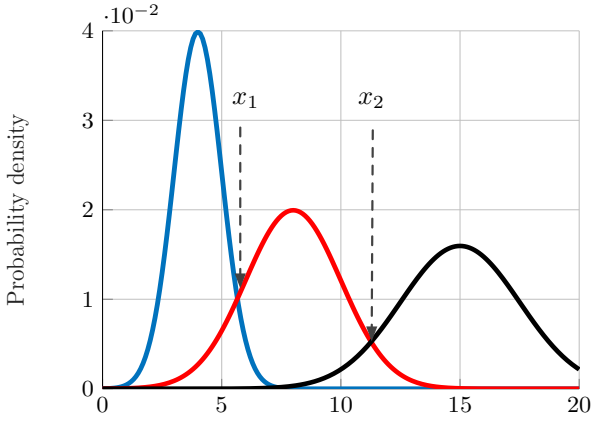


Figure 4: Thresholds selection in the case of three normal probability distribution functions

2.4. Path Cost Function

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

$$g(s,s') = \begin{cases} \frac{\sqrt{(\Delta x_r)^2+(\Delta y_r)^2}}{\dot{\boldsymbol{r}}^+} \times DF & \text{if } \Delta x_r \neq 0 \text{ or } \Delta y_r \neq 0 \\ \frac{\Delta\theta_r}{\dot{\theta}^+} \times DF & \text{otherwise} \end{cases} \tag{4}$$

where Δx_r, Δy_r and $\Delta\theta_r$ are the differences between the x_r, y_r and θ_r, which are the coordinates of the robot's pelvis joint, between states s and s', DF is a difficulty factor associated with each primitive, $\dot{\boldsymbol{r}}^+$ is the maximal robot linear velocity and $\dot{\theta}^+$ is the maximal angular velocity for turning in place. The Euclidean distance between both states is computed and then divided by the maximum velocity of the robot in the direction of the movement to give the approximate time to execute the primitive. Since the robot usually slows down when approaching obstacles to avoid collisions, transitions that pass close to obstacles have higher costs.

For both instances, the time cost estimate is then multiplied by the DF associated with each primitive. This DF is used to prioritize or penalize certain motions or directions, which result in a smoother and a more natural-looking trajectory. For instance, turning in place then moving forward takes a longer time than moving in diagonal. 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 distance, it is more likely that turning in place and moving forward would be preferred, examples of DF values are given in Table 1. Note that our experiments with the Nao robot

showed that the admissible angles for turn in place motion are in the interval $[-\frac{\pi}{3}, \frac{\pi}{3}]$ with a sampling of 0.196 rad for the omnidirectional set and in $[-\frac{\pi}{8}, \frac{\pi}{8}]$ with a sampling of 0.098 rad for the heavy set.

	Omnidirectional set	Heavy set
Forward	1	1
Turn in place by angle θ (in rad)	$1+\frac{3*\lvert\theta\rvert}{\pi}$	$1+\frac{8*\lvert\theta\rvert}{\pi}$
Backward	3	N/A
Sideways	2	N/A
Diagonal	1	N/A

Table 1: The DF factor for different motion classes of each set.

2.5. Search Algorithm

A* is one of the most popular search algorithms to find an optimal solution path using a cost function. In addition to the path cost function, a heuristic bias the search towards the most promising states. Even though A* is optimal when it finds a solution, that solution may, however, not always exists or cannot be found within a reasonable time. The Anytime Repairing A* (ARA*) planner focuses on delivering a suboptimal solution as fast as possible; this solution is then optimized iteratively to obtain the optimal solution within a predefined limited time. Also, the states are expanded in the opposite way, from goal to start, so that the heuristic costs remain valid after replanning and do not need to be recomputed. Since the algorithm is a time constrained sub-optimal A*, it also guarantees completeness. The cost function takes the form of:

$$f(s, s') = g(s, s') * max(Cost_{cells}(s, s')) + \epsilon h, \epsilon >= 1 \tag{5}$$

where $g(s, s')$ is the path cost of equation (4), h the heuristics that uses a grid of 2D distance cost computed with a Dijkstra search from the goal to the start states and

$$Cost_{cells}(s, s') = \begin{cases} 1 & \text{free space} \\ 2 \text{ to } 99 & \text{inflation} \\ \infty & \text{obstacles} \end{cases} \tag{6}$$

a vector containing the cost of every cell between s and s'. The search is biased towards states that are closer to the goal and return a solution that is, at worst, ϵ times the cost of the optimal solution providing user defined bounds on the sub-optimality of the solution.

3. Controlling the robot and cart-like object

3.1. Object Stability and the Hand Stabilization

Since the cart is fully controlled by the robot's hands, it is possible to reduce the lateral swing created by the humanoid walk. In addition, the load stability

is improved by restraining the propagation of these oscillations to the table. At low speed, a humanoid robot has no other choice rather than to move the horizontal projection of its Center of Mass (CoM) from one support (foot) to the other in order to keep its Zero Moment Point (ZMP) within the support polygon and stay in balance. This lateral motion causes the entire upper body of the robot to oscillate laterally at an amplitude proportional to the distance between the center of feet, which in turn causes the side of the cart, which is held by the robot, to move by the same amplitude. It is usually unsafe to transport a load that continuously swing from one side to another and could even damage to the transported objects or the surroundings.

One way to compensate this instability without changing the walking gait and feet position is to use the robot's arms. To this end, the hands are kept at a fixed position with respect to the fixed frame of support foot; this position is determined at the initial starting position of the robot while holding the table and corresponds to the transformation between both feet and both hands. While moving, the transformation of the hands w.r.t. the fixed foot is used. Those transformations will be the input of the whole-body control scheme described in the following section.

3.2. Whole-body Control Scheme

Once a collision-free trajectory is found by the ARA* algorithm, a set of footprints are defined along the trajectory as it is shown in Fig. 5. The second step is to define a dynamically stable trajectory by defining an appropriate ZMP trajectory [15]. A trajectory of the CoM of the robot is then obtained using the preview control algorithm proposed in [15]. This algorithm has been widely used by researchers in humanoid robotics, it is simple to implement, yet efficient and yields a smooth CoM trajectory by minimizing the CoM jerk trajectory. The foot trajectories are obtained by spline interpolation between the footprints and the hand trajectories and orientations are defined in order to minimize the walking swing effect as well as follow the cart orientation.

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

$$
\begin{aligned}
& \min_{\dot{\boldsymbol{q}}} \dot{\boldsymbol{q}}^T \boldsymbol{Q} \dot{\boldsymbol{q}} \\
& \text{subject to} \\
& \text{First priority} \quad \begin{cases} \boldsymbol{J}_c \, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_c \\ \boldsymbol{J}_{lf} \, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{lf} \\ \boldsymbol{J}_{rf} \, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{rf} \end{cases} \\
& \text{Second priority} \quad \begin{cases} \boldsymbol{J}_{lh} \, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{lh} \\ \boldsymbol{J}_{rh} \, \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_{rh} \end{cases} \\
& \text{Joint velocity limits} \quad \hat{\dot{\boldsymbol{q}}}^- \leq \dot{\boldsymbol{q}} \leq \hat{\dot{\boldsymbol{q}}}^+
\end{aligned}
\tag{7}
$$

where $\dot{\boldsymbol{q}} \in \mathbb{R}^n$ is the joint velocity vector, $\boldsymbol{Q}$ is a positive semi-definite matrix, $\boldsymbol{J}_c \in \mathbb{R}^{3\times n}$, $\boldsymbol{J}_{lf} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{rf} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{lh} \in \mathbb{R}^{6\times n}$, $\boldsymbol{J}_{rh} \in \mathbb{R}^{6\times n}$ are the jacobian

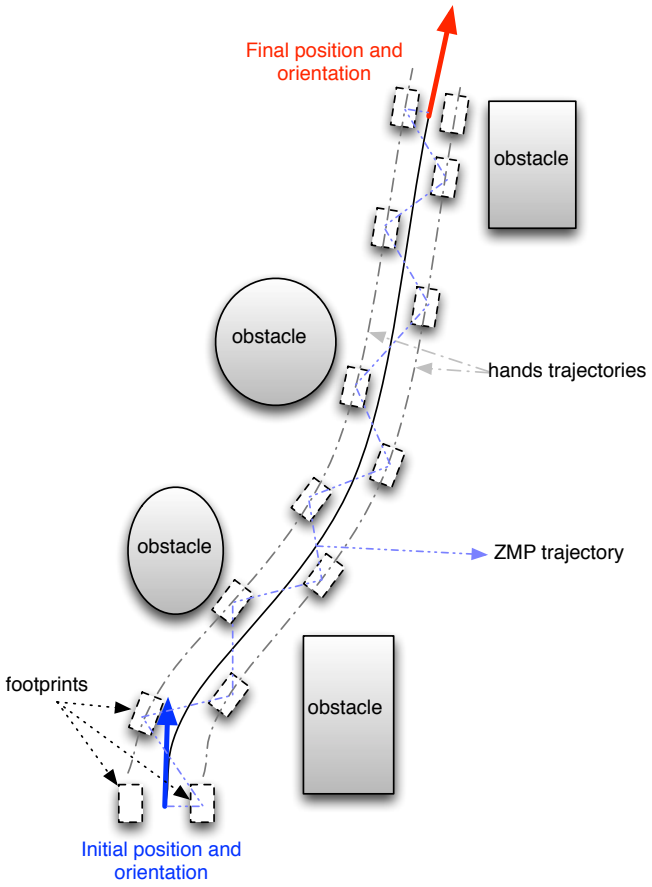


Figure 5: Overview of the motion planning procedure

matrices of CoM, left foot, right foot, left hand and right hand respectively. $\dot{\boldsymbol{r}}_c$, $\dot{\boldsymbol{r}}_{lf}$, $\dot{\boldsymbol{r}}_{rf}$, $\dot{\boldsymbol{r}}_{lh}$, $\dot{\boldsymbol{r}}_{rh}$ are the linear and angular velocity of CoM, left foot, right foot, left hand and right hand respectively.

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

$$\hat{\dot{\boldsymbol{q}}}_j^+ = \begin{cases} \dot{q}_j^+ \frac{(q_j^+ - q_j) - \rho_s}{\rho_i - \rho_s} & \text{if } q_j^+ - q_j \leq \rho_i \\ \dot{q}_j^+ & \text{otherwise} \end{cases}$$
$$\hat{\dot{\boldsymbol{q}}}_j^- = \begin{cases} \dot{q}_j^- \frac{(q_j - q_j^-) - \rho_s}{\rho_i - \rho_s} & \text{if } q_j - q_j^- \leq \rho_i \\ \dot{q}_j^- & \text{otherwise} \end{cases} \tag{8}$$

where $\hat{\dot{\boldsymbol{q}}}_j$ is the j 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, ρ_i and ρ_s are usually called the influence and security distances respectively. It can be easily proven that the equality constraints in (8), not only yield a motion within the humanoid's velocity limits, but also the joint limits are respected as well with a safety margin equals to ρ_s:

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

Eq. (8) provides a compact and efficient way to deal with both of velocity and joint limits, it has been originally proposed in [16].

The optimization problem (7) can be transformed into a standard Quadratic Programming (QP) problem as follows:

$$\begin{aligned} &\min_{\dot{\boldsymbol{q}}, w} \dot{\boldsymbol{q}}^T \boldsymbol{Q} \dot{\boldsymbol{q}} + w^T \boldsymbol{Q}_w w \\ &\text{subject to} \\ &\quad \boldsymbol{J}_1 \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_1 \\ &\quad \boldsymbol{J}_2 \dot{\boldsymbol{q}} = \dot{\boldsymbol{r}}_2 + w \\ &\quad \hat{\dot{\boldsymbol{q}}}^- \leq \dot{\boldsymbol{q}} \leq \hat{\dot{\boldsymbol{q}}}^+ \end{aligned} \tag{9}$$

where:

- w is a slack variable that is introduced to ensure the priority feature of (7).

- $\boldsymbol{Q}_w \in \mathbb{R}^{12\times 12}$ is a user-defined positive-definite matrix. In order to respect the task priority, the following condition should be satisfied: $\|\boldsymbol{Q}_w\| \gg \|\boldsymbol{Q}\|$ (in practice, we usually use $\boldsymbol{Q}_w = 10^n \times \boldsymbol{Q}$, where n could be 3, 4 or 5).

- $\boldsymbol{J}_1 = \begin{bmatrix} \boldsymbol{J}_c \\ \boldsymbol{J}_{lf} \\ \boldsymbol{J}_{rf} \end{bmatrix}$ and $\boldsymbol{J}_2 = \begin{bmatrix} \boldsymbol{J}_{lh} \\ \boldsymbol{J}_{rh} \end{bmatrix}$

- $\dot{\boldsymbol{r}}_1 = \begin{bmatrix} \dot{\boldsymbol{r}}_c \\ \dot{\boldsymbol{r}}_{lf} \\ \dot{\boldsymbol{r}}_{rf} \end{bmatrix}$ and $\dot{\boldsymbol{r}}_2 = \begin{bmatrix} \dot{\boldsymbol{r}}_{lh} \\ \dot{\boldsymbol{r}}_{rh} \end{bmatrix}$

The QP problem (9) can be solved in real time using an appropriate QP solver such as uQuadProg solver [17] or qpOASES solver [5]. Note that the above formulation supposes that the first priority kinematics task is always feasible, this is mainly true in our case. However, in a general case, a second slack variable should be added to the first priority constraints [3]. The output of the above QP problem, $\dot{\boldsymbol{q}}$, is integrated to obtain $\boldsymbol{q}$ and then used as the desired reference for the robot's joints.

4. Simultaneous Localization and Mapping (SLAM)

To move in a cluttered environment, a robust and precise sensing input is primordial to determine the position of obstacles, detect collisions and to plan valid long term and short term paths. Also, odometry drift must be constantly verified and corrected by an accurate localization mechanism to ensure the planned path is closely followed. The human-size humanoid robots, such as HRP-2 or the humanoid robots which participated in 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 only two cameras in its head for sensing and localization. A first approach would be using those cameras, however, this has proven to be a very difficult task [33] [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 is greatly obstructed by the large table and load. As a result, it is hard to precisely determine the position of the environment and obstacles with respect to the robot while only relying on the Nao's cameras.

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

Another approach would be adding a depth camera on the top of Nao's head for mapping [24]. To achieve this, an open-source library named RTAB-Map [20, 21] has been used. RTAB-Map is a RGB-D Graph SLAM library that uses a bag-of-word 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. RTAB-Map also provides a robust odometry system based on visual information. It can create 3D maps of the environment as well as constructing a 2D occupancy grid map by projecting the obstacles on the ground plane. Even though our system can use probabilistic grids, the library only provides deterministic occupancy grids at the moment. The library also supports large maps with paths of kilometers long, multi-session mapping and localization.

4.2. Filtering out the Cart

Since the cart is in the field of view of the camera, it needs to be filtered out in order to construct a valid occupancy grid or it will constantly be treated as an obstacle. The easiest solution is to simply ignore the part of the image where the cart is located. However, filtering a fixed cart position does not take into account the arms that can turn the table and the oscillation of the camera with respect to the table as shown in Fig. 6. Moreover, removing the entire zone where the cart might be located will result in ignoring the majority of the image and leads to poor performances. An efficient solution is to label all 3D points lying on planes parallel to the ground as free space.

Without filtering, that means when segmentation is true in Algorithm 1, only the ground is extracted and considered as free space, all the rest are obstacles. By taking into account that the vast majority of obstacles are bounded by edges that are not parallel to the ground, it is possible to filter out all flat surfaces

Algorithm 1 GetObstacles

```
flatSurfaces ← normalFiltering(groundNormalAngle)
if (segmentation == true) then
  cluteredSurfaces ← extractClusters(flatSurfaces)
  ground ← getBiggestCluster
  for all cluster ∈ clusteredSurfaces do
    if cluster.centroid is close to ground.centroid then
      ground += cluster
    end if
  end for
else
  ground ← flatSurfaces
end if
obstacles ← extractClusters(!ground)
```

with a normal parallel to the ground normal. This allows the robot to filter the cart no matter its position and orientation, thus using the points on its sides for localization and mapping.

The edges of obstacles generate walls around them that prevent motions to be planned over the obstacles. Noe that boxes, tables, desks and chairs with a non-zero width and their legs will all be easily detected. Only very thin surfaces with no distinguishable width and visible legs will be wrongly ignored. We believe, however, that this type of objects is not representative of typical obstacles. This filtering technique also reduces the required calculations to search for the ground by the vision library, since the clustering of flat surfaces is bypassed.

4.3. Odometry fusion

A problem that occurs when using the visual odometry produced by RTAB-Map is that it can be unreliable at times and lose track of the position for multiple reasons such as, but not limited to, lack of detected features in the observed environment, rapid movements of the camera or intense oscillations. When this occurs, the library suggested solution is to go back to where the tracking was lost. This solution is not always possible, however, even when it is possible, this is generally not an efficient and desired behavior. For this reason, a fusion of the visual odometry, the robot's internal odometry and the error between those 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 matrix between the map frame and, respectively, the robot frame as computed by the visual odometry and the camera frame. The previous equation can be rewritten to include the encoders-based robot odometry T_r^o, that does not take

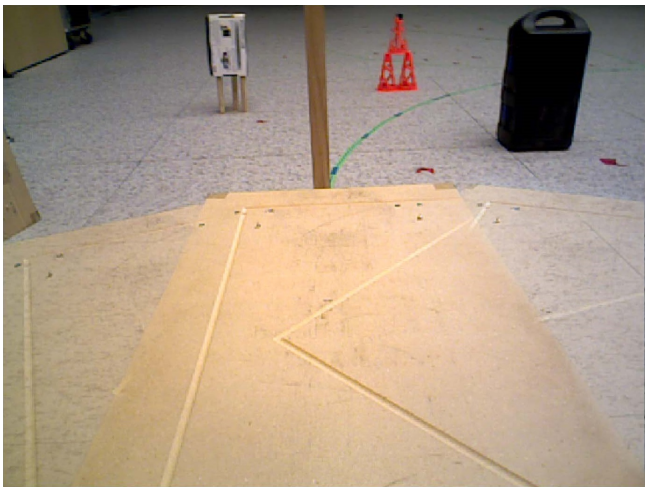


Figure 6: The position of the cart as seen by the robot. In transparency, the position of the cart when being turned at maximum angle. The images of this paper are best seen in its color version.

into account slipping, drift and other real world errors

$$\begin{aligned} T_v^o &= T_v^c * T_c^o * T_r^{o-1} * T_r^o \\ &= T_v^r * T_r^o, \end{aligned} \tag{10}$$

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 occurs at t_{lost},

$$T_v^o(t) = \begin{cases} T_v^r(t) * T_r^o(t) & \text{if } T_v^c \text{ exists,} \\ T_v^r(t_{lost}) * T_r^o(t) & \text{otherwise.} \end{cases} \tag{11}$$

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 the SLAM library and the visual odometry is restored.

4.4. Dynamic collision avoidance

A collision might occur if an obstacle has been moved or a drift from the planned trajectory happened. To check for collision, the robot and cart footprints are projected on the ground. Then, the velocity and the direction are used to interpolate the displacement and potential collisions. When a collision is foreseen, a replanning is necessary as shown in Fig. 7. Even though, at first glance, the support polygon appears to be increased by adding the cart, the robot's support polygon is always defined by the contact between the feet and the ground. This is because the robot's arms are not fully bent, therefore the robot could fall forward or backward.

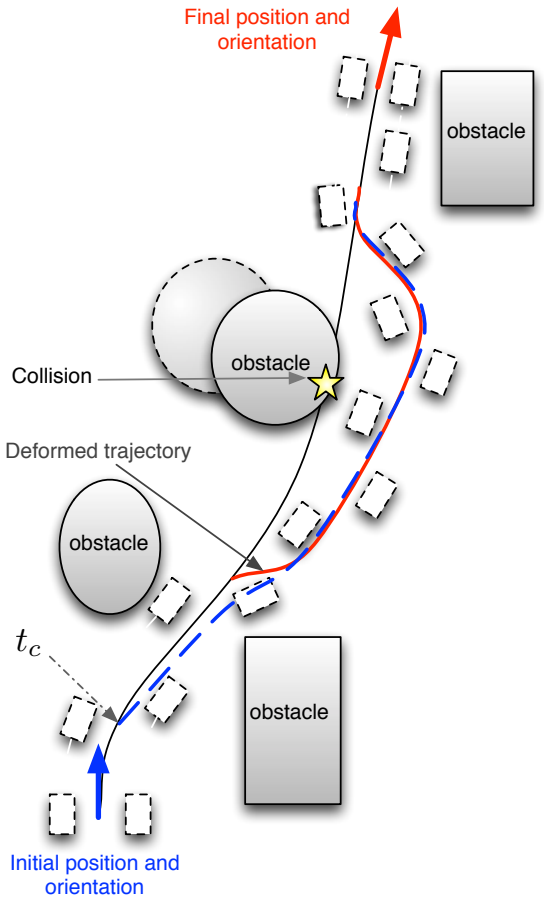


Figure 7: Replanning in case of collision detection: t_c is the instant at which a collision is foreseen, the new collision-free trajectory is in dashed-blue line, the deformed trajectory is in red line.

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 the drift, the Dijkstra grid does not need to be recalculated, therefore greatly accelerating the replanning. As the walking pattern trajectory for a humanoid robot 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 the 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. 7, the robot will, however, stop if the deformed trajectory is in collision.

5. Results

Experiments were conducted on a Nao humanoid robot (Fig. 8), manufactured by Aldebaran Robotics [7]. Its 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 each 5 DOF, while the head has 2 DOF and the pelvis and hands have 1 DOF each. In addition to the above-mentioned 25 DOF, the Nao possesses two cameras, four sonars, four force sensitive resistors under each foot, two speakers and four microphones. An IMU provides odometry data and 36 magnetic rotary encoders give joint angle information with a precision of 0.1°. On top of its head, we added an Asus Xtion Pro Live consumer-level depth camera.

Figure 8: The Nao robot holding the cart-like object and a load

The cart-like object, shown in Fig. 8, is a mini table 600mm long by 300mm wide. On one side, the two legs are 300mm high and are set on omnidirectional 40mm by 20mm ball wheels. On the other side, the two legs are half the length so that the Nao robot can fully support its side of the table.

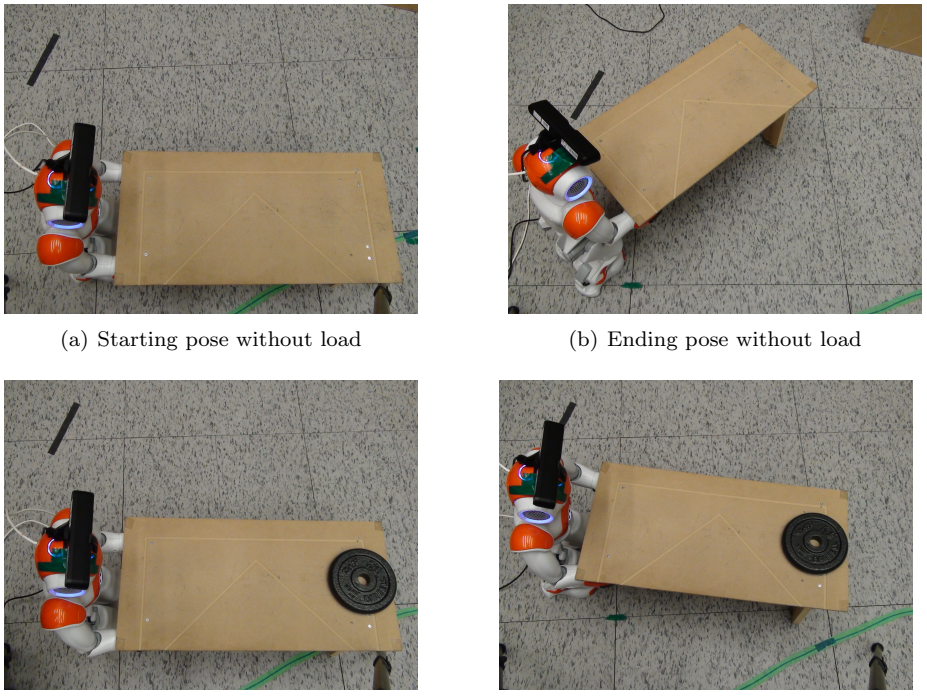
5.1. Increase in Carrying Capacity

The primary objective of our approach is to increase the maximum payload carried by a humanoid robot, without destabilizing it, while maintaining sufficient flexibility and agility. For that purpose, an experiment aimed at measuring the maximum carrying capacity of the Nao robot without any modifications has been carried out. Nao is placed in the same pose as in Fig. 8, then a small board (333g) is attached to both hands and is used to support various amount of calibration weights. Even with low weight, the robot rapidly falls down or has difficulties to follow planned trajectories. At an additional 300g weight, however, Nao falls within the first steps every run, which was determined to be its maximum carrying capacity.

With the introduction of the cart, two sets of motion primitives are available, the omnidirectional and the heavy load sets. The former generally yields shorter and faster paths than the latter, but as their names suggest, the heavy load set

allows an increased maximum load capacity. The load at which the friction becomes too high to consider the omnidirectional set is 700g. Starting from a load of 700g and up to 7,000g, excluding the cart weight of 1,370g, the heavy set must be used since Nao is only able to move forward and rotate around the wheels pivot. In reality, the robot could push higher load, however, the wood structure of our cart cannot support more than a total of 8,370g at which its structural integrity is compromised. We are, however, confident that the robot could push a higher load without that constraint. To summarize, the maximum carrying capacity of the Nao robot alone is 633g and by using the cart, it is 7,000g, which is 11 times its normal capacity.

5.2. Automatic primitive sets selection



(a) Starting pose without load

(b) Ending pose without load

(c) Starting pose with load

(d) Ending pose with load

Figure 9: Robot starting and ending poses for the automatic load estimation trial using turning in place motion.

In order to verify our approach of sensorless automatic selection of the most appropriate primitive set, we executed a series of 20 startups without load and 20 startups with a load of 2.3 kg. Multiple movement commands have been evaluated for 10 seconds per test. These movements are, walking forward in the sagittal plane (X axis), walking in the lateral direction (Y axis) and turning in

place. The latter is shown in Fig. 9 to illustrate the difference between the end poses with and without load.

It can be observed from Table 2 and Fig. 10 that turning in place by $\frac{\pi}{2}$ rad yields the most distinctive results. By approximating the probability density of distribution functions, in the case of turning in place, by normal distributions, a threshold $x = 2.645\,rad$ is obtained using (3).

Note that turning in place by $\frac{\pi}{2}$ rad does not belong to either omnidirectional or heavy primitive sets, it is only used as a test motion in order to select the appropriate primitive set. The turning in place angle ranges are given in Section 2.4.

To validate the above-mentioned threshold, we conducted another series of 10 startups, and every test finished with a successful choice of the appropriate primitives set.

	X axis	Y axis	Rotate in place
No load			
Linear error (m)	0.172	0.127	0.146
Linear error (std)	0.013	0.024	0.043
Angular error (rad)	0.091	0.073	2.103
Angular error (std)	0.042	0.038	0.124
2.3kg load			
Linear error (m)	0.176	0.146	0.085
Linear error (std)	0.018	0.016	0.031
Angular error (rad)	0.072	0.223	3.022
Angular error (std)	0.041	0.052	0.078

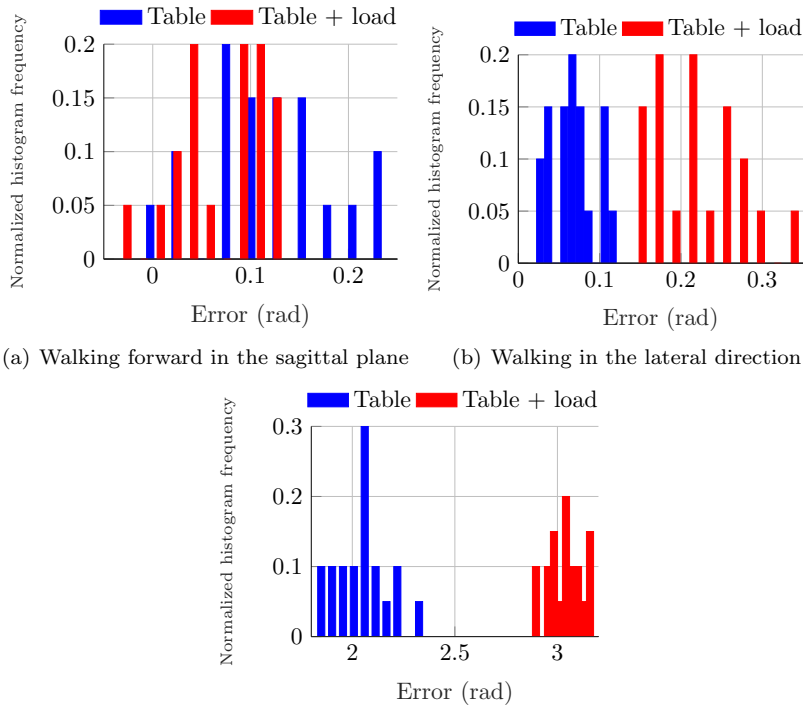
Table 2: Linear and angular errors, with and without a load, for all three basic motion candidates (std stands for standard deviation)

5.3. Articulating the arms

In the case of omnidirectional set, the hands and arms are strong enough to articulate the cart while turning to obtain smooth trajectories. The maximum angle at which our robot can turn the cart is 30 degrees, as illustrated in Fig. 11. Over that limit, one hand is colliding with the torso while the other lies outside of the robot workspace.

While walking in a straight line of 1 m, the Nao robot is affected by significant linear and angular drift. The amount of drift differs depending on whether the robot is walking alone, is using the cart or if there is a load on the cart. This information is summarized in Table 3. Without any corrections, this error would lead the robot to constantly diverge from the planned trajectories. The odometry fusion technique explained in Section 4.3 is therefore used to reduce the natural drift of the robot odometry alone.

While pushing heavy load, the robot cannot, however, rotate in place or move laterally to cancel any drift errors that accumulate over time; a quick



(a) Walking forward in the sagittal plane

(b) Walking in the lateral direction

(c) Turning in place by $\frac{\pi}{2}$ rad

Figure 10: Normalized histograms of the trials angular errors

	No Cart	With Cart	Cart & Load
Angular drift (°)	12.67	12.75	9.33
Linear drift (cm)	22.03	3.85	11.47

Table 3: Drift affecting a Nao robot while walking in a straight line of 1 m

replanning is therefore executed when the robot diverges too much from the planned trajectory.

As shown in Fig. 12(a), 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 3.2. However, as it can be seen in Fig. 12(b), the error is not completely canceled, this is mainly because: I) the hand trajectory are second priority tasks, II) Nao has only 5 DOF in each arm, III) the backlash of Nao motors, IV) the time response of the motors. To minimize the impact of the backlash and the motor-time response, a PID (Proportional-Integral-Derivative) controller has been implemented in the arms

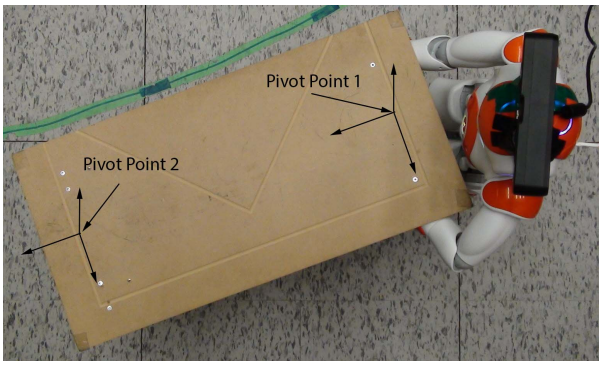


Figure 11: The arms posture while turning the cart at maximum angle (30 degrees)

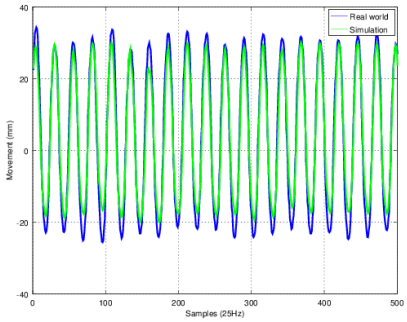
joint trajectory tracking controllers. Since the PID role is local by minimizing the error between the desired and the executed positions, it does not interfere with the formulation and the performance of the whole-body control scheme. To verify the efficiency of the hands control on swing reduction, it has firstly been tested on a simulated Nao robot.

Without any corrections, the average peak-to-peak position movement of the hands is 47.6 mm. However, with the whole body control, the hand distance from desired position has been reduced to 16.4 mm, reducing the hand error by 65.5%. By adding the PID controller with coefficients Kp = 2, Ki = 0.01 and Kd = -0.015, the error has been reduced further to 76.1% for an average peak-to-peak movement of 11.4 mm. As a result, significantly less oscillations are transmitted to the table, leading to a safer and enhanced carrying ability and load stability.

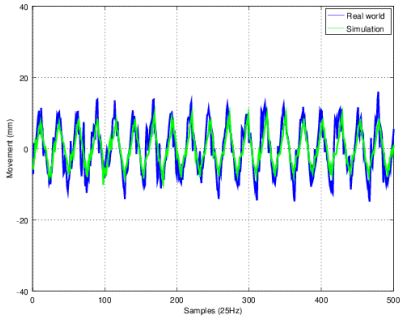
The same tests were then conducted on the real robot. The results of the simulated and real-world test are summarized in Table 4. It can be seen that the results are quite similar, thus the proposed technique significantly increases the load stability.

	Without Corrections	No PID	With PID
Simulated			
Peak-to-peak movement (mm)	47.6	16.4	11.4
Improvement (%)	0	65.5	76.1
Real world			
Peak-to-peak movement (mm)	56.0	23.3	15.6
Improvement (%)	0	58.5	72.1

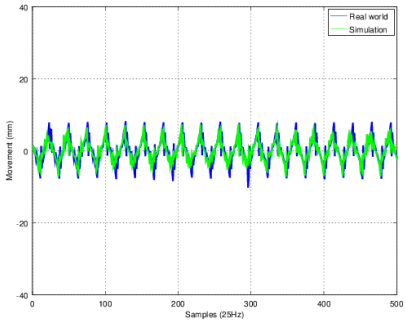
Table 4: Simulated and real world hands corrections improvements



(a) No hands correction



(b) Position control correction



(c) Position control and PID correction

Figure 12: Influence of hand corrections on table oscillations.

5.4. Navigating in a cluttered environment

To test the system as a whole and to validate the proposed algorithms, we conducted three series of 5 experiments. In each experiment, the robot starting and goal positions were chosen in a way that the robot had to navigate through a field of objects on the ground. Every one of them served as obstacles that must be avoided by the robot and the cart. They were placed to form various feasible paths and force tight turns in order to take advantage of the additional degree of freedom (the rotation of the cart θ_{ca}). The three series were composed of the same experiments containing the same initial configuration of the obstacles in the environment and using the same initial and goal positions and orientations, but with different transported objects.

Fig. 13 shows the start and end positions for each type of experiment. In this figure, the obstacles are in black, while the red areas around them are inflation zones where the cost is higher than free space to prevent the robot from passing too close to obstacles. These zones are used as a security buffer and the center of the robot and the cart should avoid, if possible, planning to pass inside it. The cyan zone is a lethal zone, because if the center of the cart or the robot enters it, it means that an edge is in collision with an obstacle.

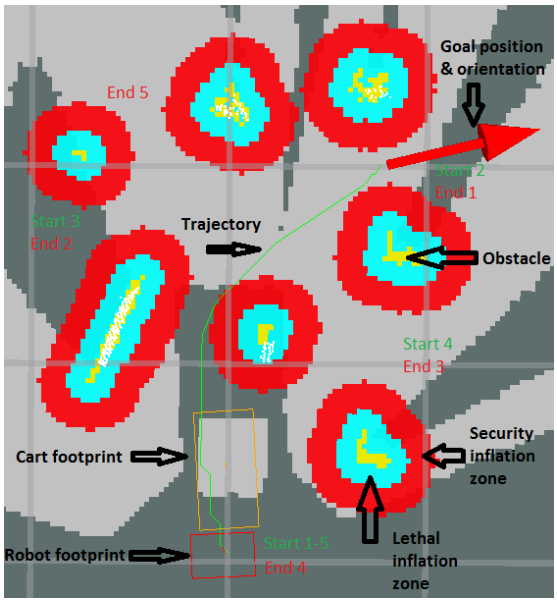


Figure 13: Map of the starting and ending positions with obstacles, lethal and security inflation zones around them, the Nao and cart footprints and goal position/orientation.

The first series consists of the Nao robot alone, without a cart or load. It chose the omnidirectional primitives set to navigate through the obstacles. The second one was conducted with the Nao holding the cart, which significantly increased the navigation footprints. These tests also chose the omnidirectional set to build the plan. For the third and final series, the robot is pushing the cart with an additional load of 2,300g, in this case the heavy load set of primitives have been chosen to allow the robot to properly plan a trajectory in the cluttered environment.

The ARA* planner initial $\epsilon = 3$ means that the suboptimal solution cannot be worse than 3 times the cost of the optimal solution. A time limit of 10 seconds was chosen and within that time, the planner converged to the optimal solution every run (ϵ successfully decreased to 1). For each generated path, we measured the total time to execute the trajectory, the trajectory length, the initial solution time, the optimal solution time, the initial expanded nodes and the final expanded nodes. These results are summarized in Table 5.

Even though all trajectory lengths are very similar, the time needed to complete the trajectory is greater with the cart than without it, and even greater with a load. This is explained by the friction on the cart wheels causing slippage as the robot tries to move and slowing its movements down. Every step in a direction results in a slippage in the opposite direction, thus progressing less distance with each step. This also holds true while rotating in place. Increasing the load weight amplifies this effect of slippage, slowing the Nao down even more. This leads to a reduced speed of 31% compared to the empty cart and

	No Cart	With Cart	Cart & Load
Total time (s)	61.04	85.82	137.28
Total time std (s)	4.91	9.46	17.76
Trajectory length (m)	2.27	2.21	3.01
Trajectory length std (m)	0.08	0.26	0.29
Goal position accuracy (m)	0.021	0.035	0.043
Goal position accuracy std (m)	0.014	0.018	0.029
Goal orientation accuracy (rad)	0.053	0.081	0.107
Goal orientation accuracy std (rad)	0.032	0.059	0.062
Average velocity (m/s)	0.0373	0.0264	0.0214
Initial solution time ($\epsilon = 3$) (ms)	2	7	19
Initial solution time std (ms)	4	8	28
Optimal solution time ($\epsilon = 1$) (ms)	173	181	214
Optimal solution time std (ms)	157	186	171
Initial node expansions	58.0	223.0	1829.6
Initial node expansions std	52.2	363.6	2103.3
Total node expansions	5711	6424	16532
Total node expansions std	7374	8648	16427

Table 5: Experimental statistics

42% speed reduction with the additional 2.3 kg load.

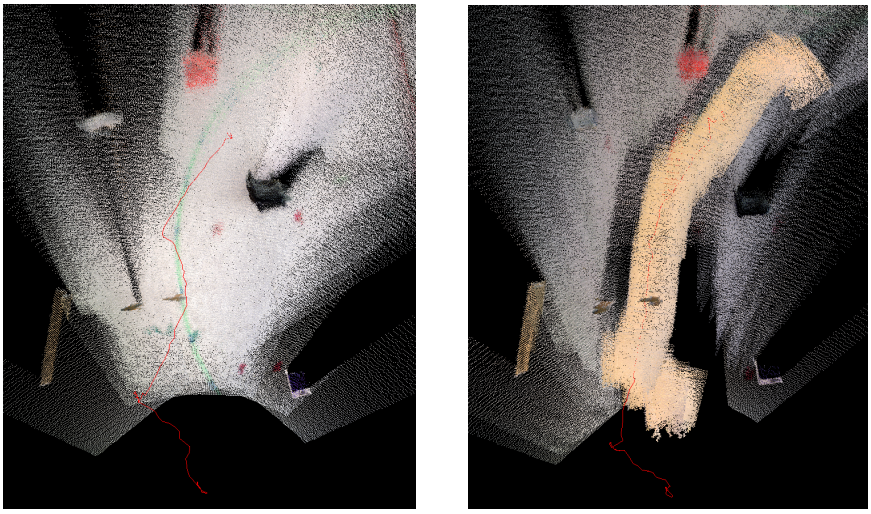
Although the primitives with the robot alone and with the cart are the same, it is hard to find a path as optimal with it. Some places might not be accessible or Nao might need more time to clear obstacles before turning because the cart is in the way. However, since the average length is about the same though, moving with the table does not impair much of the robot ability to travel the cluttered environment swiftly.

The weight primitives trajectory length is, however, higher in comparison, about 29% higher than the Nao alone. This is primarily due to the change of primitives. Moreover, some motions such as moving sideways or in diagonal become impossible. Also, turning in place by θ rad versus turning around a pivot placed $r = 0.60m$ away increased the trajectory by at least $L = \theta r$ every time the robot makes a turn.

Table 5 also shows the impact of the load on the accuracy of goal position and orientation, and it is important to point out that a threshold on that accuracy should be defined by the user, otherwise the robot will continue making small movements and never stop.

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 takes about 83, 28.7 and 11.1 times less states expansions and is 99.3, 29.1 and 9.4 times faster to compute the first solution for $\epsilon = 3$ as opposed to the optimal solution for the robot alone, with an empty cart and with a load respectively. This could be especially useful for real life large scale distances and experiments where quick reactions and planning are necessary.

The output from the SLAM library is shown in Fig. 14. The localization information is displayed as a continuous red line and the mapping information is represented by the rest of the point cloud built incrementally. It can be observed that mapping, while pushing the table, obstructs the ground and thus yields a less practical and complete map of the environment. However, since it is possible to use the side of the table for localization thanks to our filtering technique, the localization data remained continuous and reliable for all the tests. It is important to note that the ground in the testing room had sufficient textures and patterns to produce reliable data using the SLAM library. When tested on a plain ground, RTAB-Map could not, however, extract enough features to use visual localization with only the obstacles.



(a) Localization (red line) and mapping (point cloud) for the robot alone.

(b) Localization (red line) and mapping (point cloud) for the robot with the table.

Figure 14: Localization and mapping

Also, the friction on the wheels is not high enough to completely prevent them from sliding and thus to act as a perfect pivot while pushing a heavy load. This caused additional errors when turning and walking, and forcing a replanning when the accumulated error became significant. For this reason, with the heavy load, 3 replanning were needed on average, while no replanning was necessary with or without the cart when no load was carried.

When an obstacle is moved, a replanning is essential to avoid collision. As shown in Fig. 15, Fig. 16 and Fig. 17, a new path is recomputed before any collisions occur and the Nao continues its way around. In all our experiments, neither the Nao nor the cart has collided with any obstacle.

6. Conclusion and Future Work

In this paper, we proposed a system capable of carrying a heavy load on an articulated cart 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 two different sets of primitives to plan a trajectory: a low to medium weight omnidirectional set and a more restrictive heavy weight set. We also proposed a sensorless technique to test a difficult movement during the startup and accurately determine the appropriate set of primitive to be used, it is based on the error between the desired and the executed commands.

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 the robot odometry to provide a consistent, continuous and reliable odometry data. While mapping the environment, the cart pushed by the robot is filtered and ignored by the library to prevent it from being considered as a permanent obstacle. 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 [13] for planning and obstacle detection. Another option could be considering the perception system proposed in [4], that system continuously integrates stereo imagery to build a consistent 3D model of the terrain which is then used to plan the robot footsteps, however, that would require upgrading the hardware of the Nao robot as it does not have a stereo vision system. It is worth noting that since the project is mainly in 2D, for the moment, the integration of the environment 3D model was not performed.

Moreover, by controlling the hands adequately using a whole-body control scheme coupled with a PID, we were able to articulate the cart in tight turns and to significantly reduce the lateral swing from propagating to the load. Fig. 15, Fig. 16 and Fig. 17 show snapshots of the Nao navigating with the different possible configurations, i.e. without cart, with the cart only and with the cart supporting a heavy load, while successfully avoiding moving obstacles.

In future works, in order to make this project completely autonomous, it is necessary to implement it entirely on the Nao itself to be processed by its internal CPU. The biggest challenge is the limited processing power of the platform in comparison to other high-end humanoid robots that often have multiple onboard computers. The possible solutions are either to change the computing hardware to increase its processing power, or to modify the algorithms to be computationally efficient and optimized for the current platform. The most demanding part of the system for the CPU, without a doubt, is the processing of the vision based odometry. Using different parameters in the library to decrease the precision and frequency of the mapping and localization would lead to improve the performance regarding the computing power, but would necessarily have a negative impact on the results.

Moreover, the friction on the wheels and the transported mass have not been taken into account in the current study. An ongoing work focuses on the

integration of two techniques [10, 11] that we have recently developed into the framework in order to consider the friction and transported load, we expect that the new pattern generator will allow us to increase the transported load while ensuring the robot equilibrium.

Furthermore, integrating a hierarchical quadratic programming scheme [3] to solve the whole-body control problem will also be investigated.

Acknowledgment

This research is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) under the grant RGPIN-2012-419406.

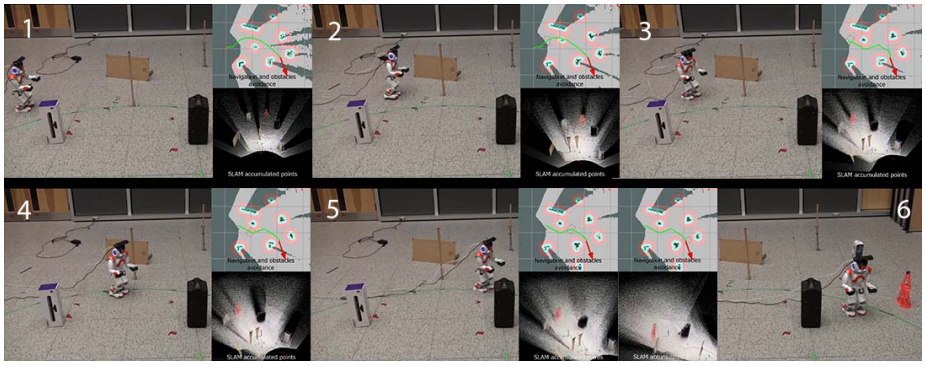


Figure 15: Snapshots of the Nao navigating without the cart using omnidirectional primitives

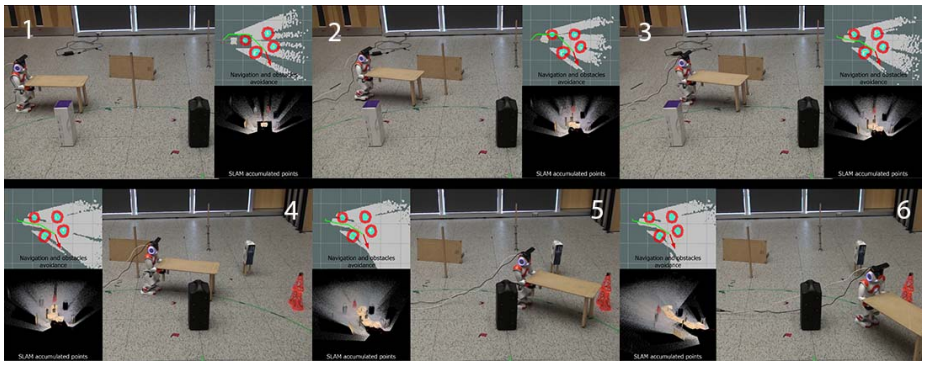


Figure 16: Snapshots of the Nao navigating with the cart using omnidirectional primitives

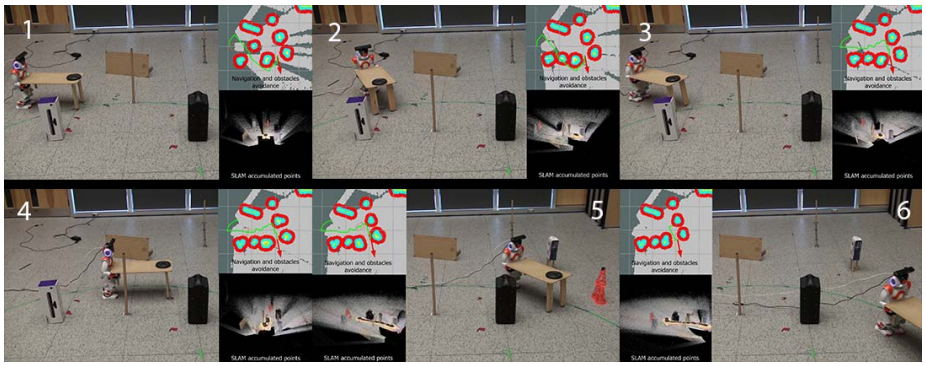


Figure 17: Snapshots of the Nao navigating with the cart using heavy load primitives

7. References

[1] Aiyama, Y., Inaba, M., Inoue, H., 1993. Pivoting: A new method of graspless manipulation of object by robot fingers. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Vol. 1. pp. 136–143.

[2] Dubins, L. E., 1957. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. American Journal of mathematics, 497–516.

[3] Escande, A., Mansard, N., Wieber, P.-B., Jun. 2014. Hierarchical quadratic programming: Fast online humanoid-robot motion generation. The International Journal of Robotics Research 33 (7), pp. 1006–1028.

[4] Fallon, M. F., Marion, P., Deits, R., Whelan, T., Antone, M., McDonald, J., Tedrake, R., Nov 2015. Continuous humanoid locomotion over uneven terrain using stereo fusion. In: 2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids). pp. 881–888.

[5] Ferreau, H., Kirches, C., Potschka, A., Bock, H., Diehl, M., 2014. qpOASES: A parametric active-set algorithm for quadratic programming. Mathematical Programming Computation 6 (4), pp. 327363.

[6] Flash, T., Hochner, B., 2005. Motor primitives in vertebrates and invertebrates. Current opinion in neurobiology 15 (6), 660–666.

[7] Gouaillier, D., Hugel, V., Blazevic, P., Kilner, C., Monceaux, J., Lafourcade, P., Marnier, B., Serre, J., Maisonnier, B., 2009. Mechatronic design of NAO humanoid. In: IEEE International Conference on Robotics and Automation. pp. 769–774.

[8] Harada, K., Kajita, S., Saito, H., Morisawa, M., Kanehiro, F., Fujiwara, K., Kaneko, K., Hirukawa, H., 2005. A humanoid robot carrying a heavy object. In: IEEE International Conference on Robotics and Automation. pp. 1712–1717.

[9] Hart, C. B., Giszter, S. F., 2010. A neural basis for motor primitives in the spinal cord. The Journal of Neuroscience 30 (4), 1322–1336.

[10] Hawley, L., Suleiman, W., 2016. External Force Observer for Medium-sized Humanoid Robots. In: IEEE-RAS 16th International Conference on Humanoid Robots. pp. 366–371.

[11] Hawley, L., Suleiman, W., 2017 (Accepted). Control Strategy and Implementation for a Humanoid Robot Pushing a Heavy Load on a Rolling Cart. In: IEEE International Conference on Intelligent Robots and Systems (IROS).

[12] Hirata, Y., Kosuge, K., 2000. Distributed robot helpers handling a single object in cooperation with a human. In: IEEE International Conference on Robotics and Automation. Vol. 1. pp. 458–463.

[13] Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., Burgard, W., April 2013. OctoMap: An efficient probabilistic 3D mapping framework based on octrees. Autonomous Robots 34 (3), 189–206.

[14] Inamura, T., Okada, K., Tokutsu, S., Hatao, N., Inaba, M., Inoue, H., 2009. HRP-2W: A humanoid platform for research on support behavior in daily life environments. Robotics and Autonomous Systems 57 (2), 145 – 154.

[15] Kajita, S., Kanehiro, F., Kaneko, K., Fujiwara, K., Harada, K., Yokoi, K., Hirukawa, H., 2003. Biped Walking Pattern Generation by using Preview Control of Zero-Moment Point. In: Proc. IEEE International Conference on Robotics and Automation. Taipei, Taiwan, pp. 1620–1626.

[16] Kanehiro, F., Lamiraux, F., Kanoun, O., Yoshida, E., Laumond, J.-P., June 2008. A Local Collision Avoidance Method for Non-strictly Convex Objects. In: 2008 Robotics: Science and Systems Conference. Zurich, Switzerland.

[17] Kanehiro, F., Morisawa, M., Suleiman, W., Kaneko, K., Yoshida, E., 2010. Integrating geometric constraints into reactive leg motion generation. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 4069–4076.

[18] Kaneko, K., Kanehiro, F., Kajita, S., Hirukawa, H., Kawasaki, T., Hirata, M., Akachi, K., Isozumi, T., 2004. Humanoid Robot HRP-2. In: IEEE International Conference on Robotics and Automation. pp. 1083–1090.

[19] Kube, C. R., Bonabeau, E., 2000. Cooperative transport by ants and robots. Robotics and autonomous systems 30 (1), 85–101.

[20] Labbe, M., Michaud, F., 2014. Online global loop closure detection for large-scale multi-session graph-based slam. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2661–2666.

[21] Labbe, M., Michaud, F., 2014. Rtab-map project on ros.org.
URL `http://wiki.ros.org/rtabmap`

[22] Lawitzky, M., Mortl, A., Hirche, S., 2010. Load sharing in human-robot cooperative manipulation. In: IEEE RO-MAN. pp. 185–191.

[23] Likhachev, M., Gordon, G. J., Thrun, S., 2003. ARA*: Anytime A* with provable bounds on sub-optimality. In: Advances in Neural Information Processing Systems. p. None.

[24] Maier, D., Hornung, A., Bennewitz, M., Nov 2012. Real-time navigation in 3D environments based on depth camera data. In: 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids). pp. 692–697.

[25] Miyata, N., Ota, J., Aiyama, Y., Sasaki, J., Arai, T., 1997. Cooperative transport system with regrasping car-like mobile robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Vol. 3. pp. 1754–1761.

[26] Nozawa, S., Maki, T., Kojima, M., Kanzaki, S., Okada, K., Inaba, M., 2008. Wheelchair support by a humanoid through integrating environment recognition, whole-body control and human-interface behind the user. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 1558–1563.

[27] Ohmura, Y., Kuniyoshi, Y., 2007. Humanoid robot which can lift a 30kg box by whole body contact and tactile feedback. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 1136–1141.

[28] Oßwald, S., Hornung, A., Bennewitz, M., 2010. Learning reliable and efficient navigation with a humanoid. In: IEEE International Conference on Robotics and Automation. pp. 2375–2380.

[29] Ota, J., Miyata, N., Arai, T., Yoshida, E., Kurabatashi, D., Sasaki, J., 1995. Transferring and regrasping a large object by cooperation of multiple mobile robots. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Vol. 3. pp. 543–548.

[30] Rioux, A., Suleiman, W., 2015. Humanoid navigation and heavy load transportation in a cluttered environment. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2180–2186.

[31] Scholz, J., Chitta, S., Marthi, B., Likhachev, M., 2011. Cart pushing with a mobile manipulation system: Towards navigation with moveable objects. In: IEEE International Conference on Robotics and Automation. pp. 6115–6120.

[32] Shigemi, S., Kawaguchi, Y., Yoshiike, T., Kawabe, K., Ogawa, N., 2006. Development of New ASIMO. Honda R and D Technical Review 18 (1).

[33] Stasse, O., Davison, A. J., Sellaouti, R., Yokoi, K., 2006. Real-time 3D SLAM for humanoid robot considering pattern generator information. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 348–355.

[34] Stilman, M., Kuffner, J., December 2004. Navigation among movable obstacles: Real-time reasoning in complex environments. In: IEEE International Conference on Humanoid Robotics (Humanoids). Vol. 1. pp. 322 – 341.

[35] Suda, R., Kosuge, K., 2002. Handling of object by mobile robot helper in cooperation with a human using visual information and force information. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. Vol. 2. pp. 1102–1107.

[36] Wang, Z., Admadabadi, M. N., Nakano, E., Takahashi, T., 1999. A multiple robot system for cooperative object transportation with various requirements on task performing. In: IEEE International Conference on Robotics and Automation. Vol. 2. pp. 1226–1233.

[37] Yamashita, A., Arai, T., Ota, J., Asama, H., 2003. Motion planning of multiple mobile robots for cooperative manipulation and transportation. IEEE Transactions on Robotics and Automation 19 (2), 223–237.

[38] Yoshida, E., Blazevic, P., Hugel, V., Yokoi, K., Harada, K., 2006. Pivoting a large object: whole-body manipulation by a humanoid robot. Applied Bionics and Biomechanics 3 (3), 227–235.

[39] Yoshida, E., Esteves, C., Belousov, I., Laumond, J.-P., Sakaguchi, T., Yokoi, K., 2008. Planning 3-d collision-free dynamic robotic motion through iterative reshaping. IEEE Transactions on Robotics 24 (5), 1186–1198.

[40] Yoshida, E., Poirier, M., Laumond, J.-P., Kanoun, O., Lamiraux, F., Alami, R., Yokoi, K., 2010. Pivoting based manipulation by a humanoid robot. Autonomous Robots 28 (1), 77–88.