Robot Grasp Planning in Domestic Environments

João Gonçalves
joaocabogoncalves@tecnico.ulisboa.pt
Instituto Superior Técnico, Universidade de Lisboa, Lisboa, Portugal
November 2018

***Abstract*—Robotic grasping in domestic environments is nowadays a topic of intensive research, since the big number of variables and constraints in a household scenario makes it a difficult task to accomplish. This thesis addresses the problem of grasp planning in these situations. The first problem to deal with is the selection of the grasp pose for the end-effector - the position and orientation in which it should grasp the desired object. To tackle this issue, this work applies a method to detect grasp poses in point clouds with objects that may be unknown to the robot, followed by an approach to select the grasp candidate in terms of its appropriateness for a given scene. The second part of the grasping task is the motion planning that places the end-effector in the grasp pose. This involves getting a Inverse Kinematics (IK) solution for the goal arm configuration and a path in joint space for the trajectory of the arm avoiding any obstacle in the scene. This was addressed by using MoveIt! framework together with an additional tolerance method to compute feasible movements. This work was implemented in ROS, using the MBOT robot, and reports results for both the real world and the simulated environment, where a comparative study regarding the used motion planners and IK solvers was conducted. It contributes with an efficient pipeline, featuring the components mentioned above, which can plan and execute grasps with a success rate that may go up to 90%, depending on the scenario.**

***Index Terms*—Uncontrolled Scenarios, Grasp Pose Detection in Point Clouds, MoveIt!, Motion Planning, Inverse Kinematics, Collision Avoidance**

I. Introduction

ROBOTS in domestic scenarios have experienced growth worldwide with a increasing number of products being created in the recent past years. Future product prospectives point to domestic robots of higher complexity and capability, such as assistive robots for supporting the more disabled people with household chores, for instance. Therefore, robotic manipulation, and grasping more specifically, is a fundamental skill required by these service mobile robots.

Theoretically, robotic grasping can be divided into two complementary actions: the perception of the environment and object to be grasped and the planning of the motion.

Applications for service household robots call for systems that are able to cope with these unstructured environments [1] which may include challenges like: variation in object placement and pose; variation in object type and appearance; sensory variation, noise and clutter in the scene; nonrigid objects; variation in the layout of the environment.

In this work, grasp planning with incomplete knowledge about the object to be grasped is addressed to devise a grasping solution for scenarios present in domestic environments as the main goal. The objectives are: 1) To detect grasp pose for a certain surface of an object without the need of identifying the object or having the object complete 3D model - capable of dealing with uncertainty arising from noise in object recognition and from uncontrolled environments; 2) To implement a feature that allows to set some tolerance values to the goal pose per each Inverse Kinematics solution, requested to the solver; 3) To plan the trajectory for the arm that allows to reach the end effector final grasp pose avoiding obstacles (self collisions and world collisions).

This thesis is integrated in the SocRob project from ISR (Institute for Systems and Robotics) that comprises a team that participates in robotic competitions such as European Robotics League (ERL) and RoboCup@Home, which also serve as motivation for this work. The presented structure is as follows: Section II presents the related work on robotic grasping. Section III presents a theoretical background of the several methods utilized. Section IV presents the approach chosen to implement grasp planning in the MBOT robot. Section V presents the results and respective discussion relative to the simulation and to the real world experiments. And, finally, Section VI presents the conclusions of this work and refers to some possible proposals of future work.

II. Related Work

A better understanding of human grasping behavior can provide basic useful heuristics for grasp planning systems in robots. Therefore, one method to understand better the grasping action is to study the human behavior in different everyday tasks and build a taxonomy of grasp types [2]. In the different taxonomies, the number of grasps presented vary to a significant degree [3].

In [4], the author studied the required conditions under which an object can be completely restrained by a grasp type. Analyzing these conditions, it is possible to define two criteria for classifying the degree of the constraints a hand is applying to an object.

The first, form closure, is defined as a condition of complete restraint in which the grasped body can resist any external disturbance wrench, irrespective of the magnitude of the contact forces [5] and is a stronger condition than the force closure criteria, which defines a grasp as force closed if it is in equilibrium for any arbitrary forces and torques, assuming

that both the objects and the links in contact are rigid bodies [6]. One common force-closure grasp is the antipodal grasp with frictional contacts, achievable with any parallel gripper and antipodal points are a pair of points on an object surface whose normal vectors are collinear and in opposite direction [7].

In order to choose the contact points - the grasp pose (end-effector position and orientation) - researchers have studied many vision methods that get the input from the sensors and give a solution for the grasping point. It is possible to divide these methods into two big groups. The first uses object recognition and aims to build a 3-D complete model of the object, that allows to estimate a 6-D pose of the object. The second does not perform any object recognition neither needs the 3-D model.

If it is possible to obtain a 3-D model of the object, then various approaches such as the ones based on friction cones, form and force-closure, pre-stored primitives can be applied [6]. However, in practical real scenarios it is often very difficult to obtain a full and accurate 3-D reconstruction, specially if the system is dealing with new objects or cluttered scenes.

Regarding the second group of vision methods mentioned above, researchers have proposed various grasp detection approaches that do not require the full 3D model. The very significant work of Saxena was the first work to orientate its efforts towards searching feasible grasps in objects independently of their identity: [8] used a sliding window classifier in images to detect possible grasps based on local visual features available, even in cluttered scenes with parts of the object occluded. Based on this work, other approaches performed a search in the object image for specific visual features in order to detect possible grasp point [9], [10], [11].

Later work explored a deep learning approach. A challenge with deep learning that is still difficult to tackle is the large volume of training data needed. Possible solutions for this problem like adversarial learning framework and parallelized learning have been published recently [12], [13].

Motion planning is an essential tool for a robotic system with autonomy. With planning, a robot's movements can be specified with start and goal configurations. Constraints are a required mechanism to set some limits to complex motions for a robot in order to avoid collisions with the obstacles in the scene during motion. This collision detection and avoidance can be performed using different methods and algorithms based on polytopes [14]. Spatial decomposition techniques have also been used for checking collisions. To create 3D maps, mobile robots sense the environment by taking 3D range measurements. [15] implements a framework based on octrees for the representation of a three-dimensional environment, the Octomap.

But before the planning is executed there is the need to specify the goal configuration that will allow to put the end effector in the position and orientation desired. For this purpose, inverse kinematics (IK) play an important role. Many robotic joints have hard limits or are redundant by having more than six degrees of freedom, which leads to the use of numerical solvers. By using the Jacobian iteratively, it is possible to compute solutions for these cases [16].

Planning for systems with constraints and obstacles in the scene is a complex task. For this purpose, sampling-based planners have been effective at planning motions for high-dimensional systems [17], since they build a discrete representation of valid motions after performing a random exploration of the robot's configuration space.

III. Theoretical Background

This section describes the methods in the literature used in this work to perform a complete grasp planning for any class of objects in domestic uncontrolled environments.

A. Grasp Model

A grasp is no more than a set of forces applied on a body by a manipulator. Each of these forces consists of a linear component (pure force) and an angular component (pure torque) acting at a point. This force/moment pair defines a wrench w.

$$w = \begin{bmatrix} f \\ \tau \end{bmatrix} \quad f \in \mathbb{R}^3, \tau \in \mathbb{R}^3 \tag{1}$$

If several wrenches are applied on the same body the resulting net wrench can be constructed by adding all the wrench vectors after transforming every w into the same reference frame. The set of all the transformations into a common reference frame defines the grasp map G.

When using wrenches as a representation of forces and torques, it becomes necessary to model the contact between the object and the manipulator. This model will provide the wrenches produced at each contact point. When considering a contact model with friction, it must be taken into account the friction cones that are built aligned with the normal to the surface of contact. The soft-finger contact model is based on Coulombs model and the friction cone presents the following formula for each contact point in 3D:

$$FC = \left\{ f \in \mathbb{R}^4 : \sqrt{f_a^2 + f_b^2} \leq \mu f_c, f_c \geq 0, |\tau_d| \geq \gamma \left| f_c \right| \right\} \tag{2}$$

where f_a, f_b, f_c represent the force along x, y and z respectively, μ is the static coefficient of friction, τ_d is the torque magnitude along the contact normal, γ is the torsional friction coefficient.

In this work, the antipodal force-closure grasps are utilized. The force-closure criteria defines that if a grasp can resist any applied wrench (force and/or torque) it is considered to be a force-closure one. In other words, given an external wrench w_e applied to the object, there is a combination of contact forces f_r applied on each contact point by the gripper such that:

$$Gf_r = -w_e \tag{3}$$

B. Grasp Pose Detection

As a first step to execute the grasp, it is very important to detect a feasible antipodal force-closure grasp pose. An efficient method to detect grasps in unknown household objects without having to identify the object to compute its pose was used in [18].

Given a point cloud from the camera of the robot, C, a region of interest, R and a parallel type gripper G, the problem of grasp pose detection is to find one or more 6-DOF end-effector poses, h, such that a force-closure grasp will be formed with respect to some object when the gripper closes.

Algorithm 1 describes the steps to perform the grasp detection and classification, adapted from [18]:

Algorithm 1 Grasp Pose Detection and Ranking algorithm

1: $C' = PREPROCESS\_CLOUD(C)$
2: $R = SELECT\_ROI(C')$
3: $S = SAMPLE(R, G, N)$
4: $I = ENCODE(S)$
5: $H = SCORE(I)$
6: $RETURN\ H$

Step 1: The procedure starts by processing the point cloud by voxelizing, removing outliers, etc. Anything that can be done to reduce noise or errors in the point cloud should be performed.

Step 2: Secondly, a region of interest (ROI) where the algorithm should look for possible grasps is specified. This creates a workspace around the object, reducing the search space, but does not mean that it segments the object out of the background.

Step 3: The goal of the third step is to find a set of grasp candidates (6-DOF hand poses) where the best grasp might be located. The method samples N points uniformly in the ROI. Then, for each sampled point p, a local reference frame is calculated by evaluating the eigenvectors of the matrix:

$$M(p) = \sum_{q \in C \cap B_r} \hat{n}(q)\hat{n}(q)^T \tag{4}$$

where $\hat{n}$ denotes the outward pointing unit surface normal at p and B_r denotes the r-ball about the point p.

Step 4: Each grasp candidate is encoded to the classifier in terms of the geometry of the observed surfaces and unobserved volumes contained within the closing region of the gripper. To represent the 3D geometry of the object surface contained within the closing region, a multiple view representation is used to encode this volume. The voxels are projected onto three planes orthogonal to the axes of the hand reference frame and pass these to the CNN as input. For each of these three projections, three images are calculated: an averaged heightmap of the occupied points, I_o, an averaged heightmap of the unobserved region, I_u, and averaged surface normals, I_n.

$$I_o(x,y) = \frac{\sum_{z\in[1,M]} zV(x,y,z)}{\sum_{z\in[1,M]} V(x,y,z)} \tag{5a}$$

$$I_u(x,y) = \frac{\sum_{z\in[1,M]} zU(x,y,z)}{\sum_{z\in[1,M]} U(x,y,z)} \tag{5b}$$

$$I_n(x,y) = \frac{\sum_{z\in[1,M]} \hat{n}(x,y,z)V(x,y,z)}{\sum_{z\in[1,M]} V(x,y,z)} \tag{5c}$$

Step 5: The final step is ranking these candidates (through the images) using a Convolutional Neural Network (CNN) that outputs a score for each of them.

To introduce this CNN concept, an overview on neural networks is firstly presented. Usually, Artificial Neural Networks (ANNs) are described by a Multilayer Perceptron (MLP). Basically, a MLP is a feedforward network and it is normally trained using a supervised method like the back propagation algorithm where the objective is to minimize a loss function by updating the weights.

Once again the main purpose of the back propagation algorithm is to minimize an error function, such as:

$$e(i) = o(i) - d(i) \tag{6}$$

where i is the input, $e(i)$ is the error, $o(i)$ is the output and $d(i)$ is the desired output.

The loss function is usually described by the least square error method:

$$E(k) = \|e(k)\|^2 \tag{7}$$

A popular minimization procedure is based on gradient algorithms, which consists in iteratively updating the weights in order to minimize the loss function:

$$w(n+1) = w(n) - l_r \cdot \frac{\partial E}{\partial W} \tag{8}$$

where w are the weights, l_r is the learning rate and $\frac{\partial E}{\partial W}$ is the derivative of the loss function in respect to the weights.

Convolution Neural Networks are a more complex type of networks, composed of a sequence of layers that are responsible to transform the image volume into an output. So, after receiving a certain number of images as input, the different stages are responsible for encoding certain properties into the architecture. After the convolution and pooling layers the net is constituted by 1D fully-connected layers, which are basic artificial neural networks, identical to MLPs. Figure 1 shows a CNN architecture.

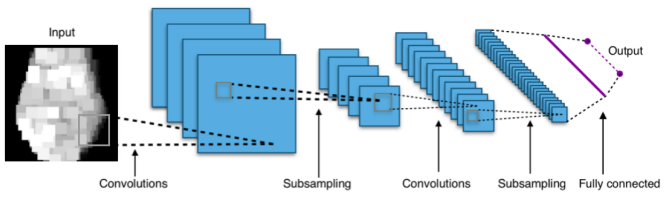


Fig. 1. CNN architecture

CNN architecture in this work: The CNN structure used in this thesis is based on the implementation by Lecun[19]: two convolutional/pooling layers followed by one inner product layer with a rectified linear unit at the output and one more

inner product layer with a softmax on the output, similar to the one in Figure 1. The outputs, kernel size, pooling strides and other parameters are all identical to those used by the LeNet solver provided in Caffe [20].

For training the CNN, a total of 300k exemplars with equal numbers of positive and negative examples for 55 household objects in the BigBird dataset [21]. These objects include 29 box-type objects, 16 cylindrical-type objects, and 10 other objects, all fitting the dimensions of the gripper. As result, during the training, each candidate was labeled as a good grasp or not by evaluating whether a force closure grasp would be formed if the fingers were to close from the given 6-DOF hand pose.

C. Motion Planning

To plan the motions of each joint of the arm, it is necessary to resort to motion planning algorithms, sampling-based ones in the case of this work. Sampling-based motion planning is a concept that employs sampling of the state space of the robot in order to quickly, yet effectively, answer planning queries, especially for systems with many degrees of freedom. This approach was intensively studied in the work of LaValle [22]. Two types of sampling-based planners should be considered: graph-based methods and tree-based methods.

Graph-based methods: The graph-based method used in this work is the Probabilistic Roadmap (PRM). It works by uniformly sampling the free state space and making connections between the samples to form a roadmap of the free state space as described in Algorithm 2. Therefore, this roadmap is a set of states in which each state is connected with all states where there is no obstacle between them. In short, PRM is computationally expensive since it builds an entire roadmap before any information is processed and only after the map is built (see Fig. 2(a)), it finds the shortest path. In case the final state is relatively near to initial state, it is not necessary to explore all map, making this method not the best choice, for this particular case.

Algorithm 2 Graph-based algorithm

```
1: Graph.init()
2: while no path from q_start to q_goal do
3:    q_rand ← SAMPLE()
4:    if q_rand ∈ Q_free then
5:       Graph.ADD_VERTEX(q_rand)
6:       for each q ∈ Graph
7:          if CONNECT(q_near, q_rand) then
8:             Graph.ADD_EDGE(q_near, q_rand)
9: RETURN Graph
```

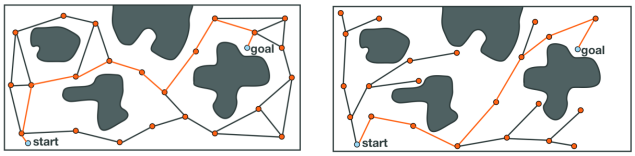
Tree-based methods: There exist many types of sampling-based planners that create tree structures of the free state space. In this work the Rapidly-exploring Random Tree (RRT), the Single-query Bi-directional Lazy (SBL) and the Kinodynamic Planning by Interior-Exterior Cell Exploration (KPIECE). These methods begin by rooting a tree at the starting configuration of the robot and exploring until the goal is reached (see Fig. 2(b)). They are suitable for single query problems, employing an exploration heuristic, which typically gives the method its name.

Algorithm 3 describes the general steps performed by these planners, where in line 4 is executed the exploration procedure that depends on each planner. There are several alternatives

Algorithm 3 Tree-based algorithm

```
1: Tree.init()
2: for i = 1 to K_iterations do
3:    x_rand ← RANDOM_STATE()
4:    x_near ← EXPLORATION(x_rand,Tree)
5:    x_new ← NEW_STATE(x_near, u, Δt)
6:    Tree.ADD_VERTEX(x_new)
7:    Tree.ADD_EDGE(x_near, x_new, u)
8: end for
9: RETURN Tree
```

for implementing the $EXPLORATION$ function. RRTs are constructed incrementally in a way that quickly reduces the expected distance of a randomly-chosen point to the tree. They are particularly suited for path planning problems that involve obstacles. SBL has the same tree expansion strategy as the Expansive-Spaces Tree planner (EST), but attempts to grow two trees at once. It detects the less explored area of the space by measuring the density of the explored space, biasing the tree exploration toward parts of the space with lowest density. Finally, the KPIECE uses a discretization (multiple levels, in general) to guide the exploration of the continuous space. A grid represents one level and is imposed on a projection of the state space.



(a) (b)

Fig. 2. Graph-based and Tree-based planners

D. Collision Detection

As a final accomplishment for the work, it is required that the planner is aware and avoids obstacles that may be in the scene. Besides this, there is also the need to represent the robot itself as an element in the scene that has to be considered when performing collision checking.

Robot applications often use a probabilistic representation of the environment because input sensors introduce noise and therefore uncertainty into the system. The used octomap framework [15] is a 3D occupancy grid mapping framework based on the octree structure. The octree data structure is a hierarchical structure containing multiple voxels (cubic volumes) that can be subdivided in child voxels. The number of subdivisions determines the precision and size of the octree (see Fig. 3).

The octomap child voxels contains a probabilistic number. This allows the octomap to have a probabilistic representation

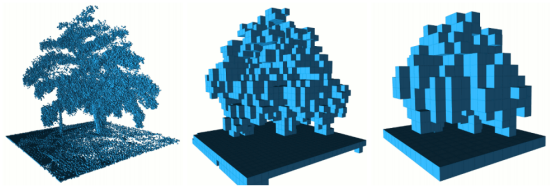


Fig. 3. Octotree with different resolutions [15]

of the environment and determine if the voxel is occupied or free. Integrating this approach means constructing a occupancy grid map. The probability P of a child node n to be occupied given the sensor measurements $z_{1:t}$ is estimated according to:

$$P(n \mid z_{1:t}) = \left[1 + \frac{1 - P(n \mid z_t)}{P(n \mid z_t)} \frac{1 - P(n \mid z_{1:t-1})}{P(n \mid z_{1:t-1})} \frac{P(n)}{1 - P(n)}\right]^{-1} \quad (9)$$

This update formula depends on the current measurement z_t, a prior probability $P(n)$, and the previous estimate $P(n \mid z_{1:t-1})$. The term $P(n \mid z_t)$ denotes the probability of voxel n to be occupied given the measurement z_t. This value is specified depending on the sensor that generated the measurement.

Regarding the representation of the robot itself as a collision element, a different approach has to be made, since the geometry of the body cannot be seen by the camera and therefore is not part of the point cloud. Instead of doing spatial decomposition, they approximate meshes of objects by a finite set of convex components named bounding volume (BV) and then they compute the possible collisions between BVs. In order to build the bounding volumes it is possible to use the Axis-aligned Bounding Box (AABB), the Oriented Bounding Box (OBB) or the Swept Sphere Volume (SSV), for instance, with the goal of constructing a BV Hierarchy represented by trees where the main node is a BV with all the object inside. Then this node is divided into children nodes, each representing a BV with a smaller part of the object inside. Using these two approaches mentioned above, a occupancy map of the environment defines the free space for the motion planner to use that checks for collision using methods based on the the Minkowski Difference.

E. IK Solver

In order to get the end-effector to the grasp pose, one must find a feasible configuration for the joints of the arm, achievable by applying inverse kinematics (IK) to the arm. The goal of IK is to compute the vector of joint DOFs that will cause the end effector to reach some desired goal state. There is a collection of possible joint angles which bring the endpoints position to the wanted goal with the desired orientation, so there is the need to compute these configurations and to choose one among all. When solving for an arm with more than six degrees of freedom, analytical methods (which are complex to solve, but faster once implemented) are not suitable, because the redundant degree of freedom makes this a problem with an infinite number of solutions (joints configurations), all of which bring the endpoints position to the wanted goal. Thus, in these cases, closed-form solution do not exist or are very difficult to compute. Therefore, numerical solvers are required and their implementation in robotic manipulation has been a field of research for some time, being specially relevant the work of [23].

The IK solution is often computed resorting to the Jacobian matrix that describes the changes of the end-effector position with respect to the changes in the joints angles. Therefore, to calculate the variation in joins angle to achieve a specific end-effector position, there is the need to invert the matrix. The relation can be described inversely by the equation:

$$\Delta\theta = J\left(\theta_{current}\right)^{-1} \cdot \Delta p \quad (10)$$

But $J\left(\theta_{current}\right)^{-1}$ is most of the times not invertible like in the case of the 7-DOF arm used in this work because the Jacobian matrix is non square. Thus, there is the need to find an alternative to cope with this problem. The most popular method is the Pseudo-Inverse [24]. To compute the Pseudo-Inverse, the Singular Value Decompositon approach is used. Using this approach, it is possible to calculate the approximate change in joint angles needed to achieve this change in position, locally and differentially, an approximation that is only valid near the current configuration. So, in order to achieve the full calculation of all the changes, several iterations are required, making this an optimization problem. A widely used algorithm for these cases is the Newton-Raphson method. However, this approach does not deal efficiently with the presence of joint limits. Therefore, it is also considered a second algorithm which provides an extension to the original Newton-based convergence algorithm that detects and mitigates local minima due to joint limits [25], using a Sequential Quadratic Programming (SQP) nonlinear optimization method.

IV. Implementation

This section presents the chosen approach for implementing the grasp planning in the MBOT, a omnidirectional robot with a 7-DOF manipulator and a description of the complete pipeline integration.

To obtain the grasp pose, it was chosen to follow the methods that do not perform object recognition or estimate its pose, but detect possible successful grasp pose candidates based only on point cloud information. These methods showed to be more efficient for real uncontrolled domestic environments and they are able to cope with novel objects or objects with parts occluded and avoid problems related with their misidentification. Regarding the motion planning, it was used a numerical solver able to give a solution for the redundant 7-DOF arm goal configuration that places the end-effector in the grasp pose avoiding obstacles together with a sample-based planner to compute the trajectory. The implementation of a pipeline that integrates the features mentioned above was executed using the Robot Operating System (ROS), MoveIt! and Rviz (for visualization purposes).

A. Robot Hardware and Software Frameworks

In this work, it was used the MBOT robot from the MOnarCH project [26]. This is a mobile robot with a omnidirectional base that allows it to move in any desirable direction.

For the manipulation actions a Cyton Gamma 1500 robotic arm was attached the robot and, for perception purposes, a Orbbec Astra S camera was attached to the top of the head of the robot (see Figure 4).

Fig. 4. MBOT robot

This robot configuration allows the robot to execute grasps with different orientations in a grasp range from the floor level to approximately 1 meter platforms. It is important to notice that in order to increase the space where the arm can perform a grasp, the robot should approach the object with its left side facing the object.

The pipeline integration was done with ROS. The fundamental concepts of ROS are nodes, messages and topics. Nodes are processes that perform computation, also known as software modules. Packages can have one or more nodes that communicate with each other by sending or receiving messages through topics.

For mobile manipulation, MoveIt! is the state of the art in ROS, incorporating the latest advances in motion planning, 3D perception, kinematics. Apart from the backend, it provides an easy-to-use GUI to configure new robotic arms with the MoveIt! and RViz plugins to develop motion planning tasks in an intuitive way. Using YAML configuration files, one can specify the motion planners, kinematics solvers, perception sensors, controllers and their parameters. Regarding the first ones, MoveIt! uses the Open Motion Planning Library (OMPL) that includes several sampling-based planners. In this work, an analysis of the behavior of these planners is made both in simulation and in real scenarios. As far as the kinematics are concerned, the Orocos KDL solver (widely used in the community) is implemented. Besides this, the Trac-IK kinematics plugin is also implemented as the extension described in section III. To detect collisions between the robot and the environment or even between parts of the robot themselves, MoveIt! uses the Flexible Collision Library (FCL) that reads the occupancy map output by the methods described in the Collision Detection section. The MoveIt! architecture is designed around a primary node called move_group. This node serves as an integrator: pulling all the individual components from the libraries and plugins mentioned above together to provide a set of ROS actions and services to be used with a group (defined by links and joints). In this case, the group is defined by the URDF file of the Cyton Gamma 1500 arm.

B. *Integration of the complete pipeline*

The implementation and integration of all functionalities together described in the beginning of this section can be seen in the Figure 5. On the left hand side, it is possible to see in red the nodes regarding the grasp pose detection and selection, exclusively a perception problem.

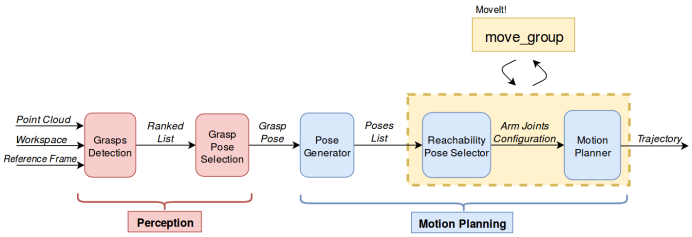


Fig. 5. Pipeline

Grasps Detection node: The pipeline gets as inputs a point cloud, a workspace shaped like a box where the method shall perform its search and the reference frame relative to the point cloud. This point cloud is given by the camera at the top of the head. The Grasps Detection node performs the method described in the Grasp Pose Detection section. This outputs a list of grasps candidates ranked by a score given by the CNN.

Grasp Pose Selection node: The Grasp Pose Selection node receives the ranked list and chooses one grasp pose to output to the motion planning part of the pipeline, allowing the integration between perception and planning. The selected grasp pose must satisfy some conditions and criteria according to the terminology adopted in Fig 6.

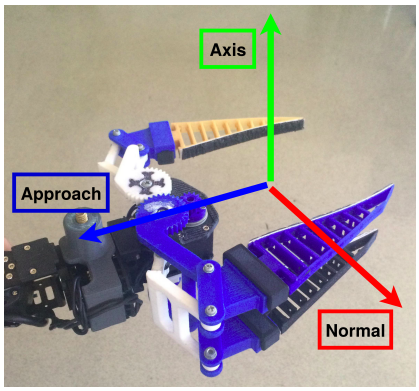


Fig. 6. End-effector frame

The required conditions are: 1) grasp pose frame should have its approach vector pointing in the 180° of the object facing the robot; 2) the axis vector should be parallel to the principal axis of rotation of the object that has the smallest moment of inertia; 3) poses too close to the platform where the object is placed are immediately filtered out.

On the right hand side of Figure 5, in blue, are represented the nodes regarding the motion planning part of the pipeline, responsible for moving the end-effector to the grasp pose selected by the perception part.

Pose Generator node: Although at this point the pipeline has already selected a goal grasp pose, there is still the need

to check if this pose is actually feasible - if there is a final arm configuration that places the end-effector in the goal grasp pose and that is not in collision with obstacles in the scene or even with the robot itself. Therefore, some tolerance must be applied to cope with these constraints. This applied tolerance will change when necessary the orientation (keeping the position) of the goal pose when comparing to the grasp pose output by the perception.

To implement this tolerance, it was developed a sampling algorithm that, given an interval of values and a sampling step, performs spherical sampling in zenith, azimuth and roll around the goal pose and with respect to the wrist link of the arm (the last link of the kinematic chain left_arm group defined in MoveIt!). Figure 7 shows the a $180°$ sampling in azimuth with a $20°$ angular step, around the green axis correspondent to the Axis vector in Figure 6.

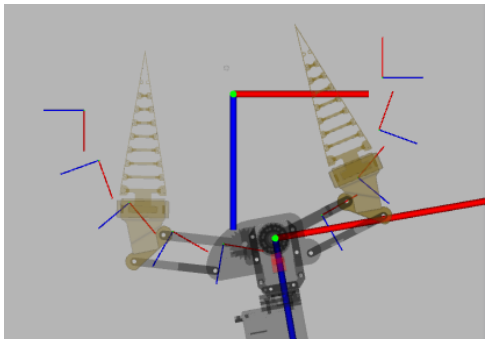


Fig. 7. Azimuth spherical sampling (-90° to 90°)

After the sampling is executed, the node outputs a list of possible poses for the wrist that places the end-effector in the goal grasp pose with a angular deviation resulting from the tolerance algorithm. It is important to notice that this list sorts the samples from the middle. In other words, the first elements of the list are the poses with an orientation closer to the original grasp pose.

Reachability Pose Selector node: This next node receives the list and starts checking for feasibility in each of the poses. Since the list is sorted from the middle, the first grasp poses to be analyzed are the ones with less angular deviation from the original one. This allows the robot to get a solution that is very similar to the pose that was output from the Grasp Pose Selection node before having to compute solutions for samples with a higher angular difference, so if a solution is found for the first samples the algorithm stops, not needing to analyze the last elements of the list. This solution computation is performed by the IK solver service provided by MoveIt!.

Motion Planner node: Finally, the Motion Planner node computes the trajectory for the arm from the current configuration to the goal configuration received. This node queries the motion planner and the planning scene services from MoveIt! to obtain a free-collision path.

V. Results and Discussion

In this section the results of the implementation in simulation and in the real world are presented.

A. Simulation

IK Solver: To test the IK solver and the tolerance algorithm, several poses were given to the solver that had to compute a solution for them. Orocos KDL and Trac-IK solvers are compared to choose the one with best performance in a simulated environment where synthetic objects were defined.

In order to get the first results to perform a preliminary analysis, it is assumed an horizontal plane at $Z = 0.5$ meters which is roughly the medium height of the arm configuration space and 95 poses in this plane equally distributed in X and Y. All the poses have the same orientation and are assumed to be oriented in a way that the Y axis simulates the main axis of an object that is standing in a synthetic table (with a height of 0.5 meters) pointing upwards vertically and the approach axis is pointing in the robot direction (according to Figure 6). Figure 8 shows in green the space where these poses were distributed. The X and Y axis shown are not meant to define the origin of the reference frame used, they are only representing the direction of each coordinate.

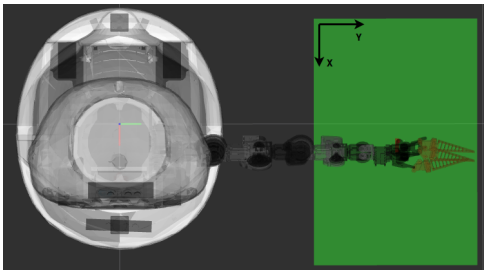


Fig. 8. Synthetic table that defines the area for the 95 simulated poses

The Orocos KDL IK solver was firstly used. The first results show the time that the Reachability Pose Selector takes to find a solution for the list of poses given by the Pose Generator. In this case, no tolerance is given, which means setting the spherical sampling values to zero, originating a list of poses with only one element. These results are presented in the table I. Note that out of the 95 poses from the total experience, it is presented here only 42 for the sake of brevity. Besides, note that the NS cells refer to No Solution cases where the iteration reached the timeout without a solution.

TABLE I
Results for no tolerance - time in seconds

y(m) / x(m)	0.60	0.70	0.80
-0.30	0.076	NS	NS
-0.25	0.055	0.059	NS
-0.20	0.040	0.027	NS
-0.15	NS	0.048	0.023
-0.10	NS	0.052	0.022
-0.05	NS	0.047	0.027
0.00	NS	0.060	0.030
0.05	NS	NS	0.030
0.10	NS	NS	0.031
0.15	NS	0.037	0.037
0.20	NS	0.044	0.033
0.25	NS	0.070	0.066
0.30	0.049	0.062	NS
0.35	0.082	0.035	NS

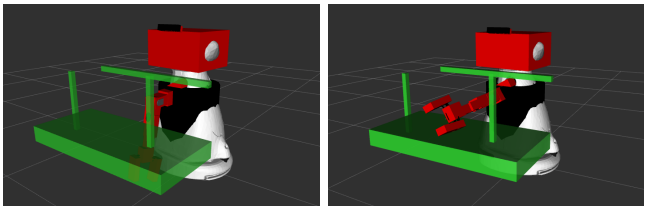
From the table, it is possible to see that for 17 out of the 42 poses, the solver was not able to compute a solution with the given parameters, representing a success rate of only 59.5%. In the cases where the solution was not found because simply there is no arm configuration feasible that places the end-effector in that pose, different options are required. By giving some angular tolerance to the goal pose with the algorithm described in section IV, one can see the results improve to a high degree. The best results were obtained as expected for the higher tolerance given (Table II): 10° deviation in zenith, 20° deviation in azimuth and 0° deviation in roll. Bigger deviations than these values are not desired since it puts at risk the success of the grasping action.

TABLE II
RESULTS FOR 10° TOLERANCE IN ZENITH AND 20° IN AZIMUTH - TIME IN SECONDS

y(m) / x(m)	0.60	0.70	0.80
-0.30	0.044	1.188	NS
-0.25	0.055	0.059	2.554
-0.20	0.045	0.064	2.114
-0.15	0.910	0.046	0.026
-0.10	1.422	0.053	0.024
-0.05	2.614	0.047	0.028
0.00	NS	0.131	0.036
0.05	NS	2.617	0.034
0.10	2.555	2.064	0.032
0.15	1.985	0.055	0.066
0.20	0.668	0.047	0.051
0.25	0.512	0.070	0.061
0.30	0.041	0.062	1.175
0.35	0.083	0.030	2.442

The results for the KDL with tolerance show a success rate of 92.9%. This shows the importance of implementing the tolerance that allows the robot to be more versatile and find solutions even when the goal pose is of difficult reachability but keeping the position of the grasping point with a very small change in the orientation of the gripper. With the results for KDL, it is possible to make a comparison with the performance of the Trac-IK. In a similar way, Trac-IK was tested with the same tolerance sampling parameters. The results show the Trac-IK performs faster, but keeping the same success rate (92.9%). However in the cases the feasible pose is close to the original goal pose (very little angular deviation) it is not evident that Trak-IK outperforms KDL. On the other hand, on the longer searches (when the search needs to search in regions close to the limits of the angular tolerance given) it is possible to see a relevant improvement in the computation time with values always lower than one second compared to the 2.6 seconds in the KDL test. To conclude, Trac-IK solver was chosen to implement in the real experiments.

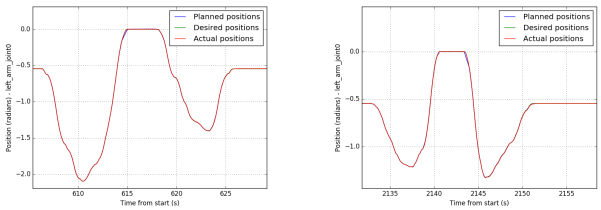
Motion Planner: To test the motion planners, the simulation environment was used, computing paths from the "kill" to the "candle" configurations. These configurations are defined in a XML file that allows to record configurations under a specific name. The success rate was measured by the number of solved runs over twenty planning requests. In Figure 9(a) is visible the "kill" configuration and in Figure 9(b) the "candle" configuration in a simulated environment. Represented in green are the synthetic obstacles arbitrarily chosen to test the four planners: PRM, SBL, RRTConnect and BKPIECE from the OMPL library. The collision model represented by the red boxes in the figures is what the planner really takes into account when checking for collision in the trajectory calculation.



(a) Kill configuration (b) Candle configuration

Fig. 9. Simulated environment

From the results of the motion planning in this simulated environment it was possible to draw some quick conclusions regrading the SBL and the PRM planners. It is important to note that the metrics used to evaluate the planners were: the computation time, the success rate of solved runs and the length of the path computed. The SBL presented a very low success rate with only 40% of solved attempts. The PRM planner, on the other hand, planned a successful path for all the twenty runs but completed each of the twenty attempts in 7.77 seconds in average. As far as the RRTConnect and the BKPIECE are concerned, there were no obvious reasons to eliminate straightaway one of them over the other since both presented a low computation time and a high success rate. The RRTConnect solved the runs in an average of 0.39 seconds with 100% of success whereas the BKPIECE solved in 1.24 seconds with 95% of successful attempts. At first, the RTTConnect outperforms the BKPIECE but an analysis of the paths computed by the two planners should me made. Figure 10 presents the results only for the path of the Joint 0 for each of the planners for the sake of brevity, but these results are representative of the rest of the joints behavior.



(a) RRT (b) BKPIECE

Fig. 10. Joints paths

From the graphs of the joints above it is possible to see that RRTConnect computes a path longer than BKPIECE since the joint "travels" 6 radians approximately compared to the 4.5 on the graph on the right. Taking all these considerations into account, the RRTConnect was chosen over the other three planners.

B. Real World Experiments

Grasp pose detection and selection: This section presents the results relative to the grasp pose detection and selection implemented in the first two nodes of the pipeline described in section IV, using the information perceived by the Orbbec Astra S depth camera. To test the method, several objects were used and divided into three categories: cylinders, boxes and others. These object were placed in a random position and orientation on a table. The results are evaluated by measuring the number of feasible grasp poses computed out of 30 runs for each object. The pose was considered to be feasible if the IK solver, along with the tolerance method, was able to compute an arm configuration for it.

In the cylinders category, the grasp pose detection and selection method was applied to a coke can (CC), a cup (C), a small pringles (SP) can and a big pringles can (standing on the table - BPS - and laying on the table - BPL). Regarding the boxes category, the grasp pose detection and selection method was applied to a big icetea box (IT) and to two medium boxes (one white - WB - and one black - BB). Finally, to test with more complex objects a bowl (B) and a duct-tape (DT) were used. Figures 11 and 12 show the results of the grasp detection and selection for the cylinders and boxes objects.

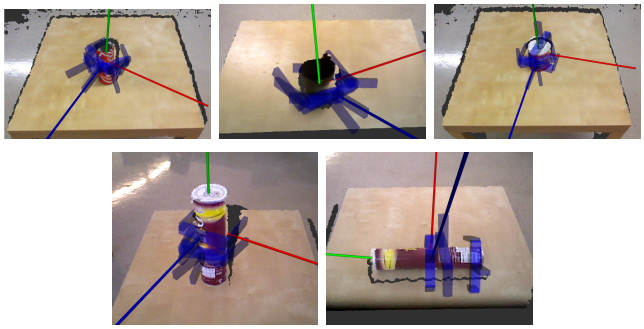


Fig. 11. Cylinders

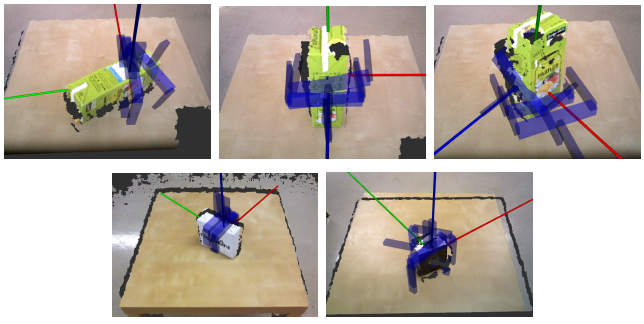


Fig. 12. Boxes

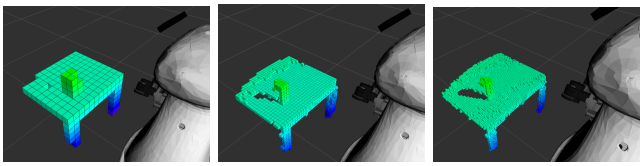
To sum up the results of this section, table III is presented.

Octomap: The dimension of these cubic volumes that represent the environment depend on the resolution that is used for the octomap. With only the coke on the table, Figure 13 shows three examples of resolutions with the cube dimensions of 5, 2 and 1 centimeters, respectively.

After testing these three resolutions, the 2 centimeters one was chosen. It is desired that the octomap represents the environment in the most precise way so the smaller the dimensions of the cubes, the higher the resolution and more precision is guaranteed. However, the 1 centimeter resolution showed to be computationally very expensive which led to time synchronization problems between the nodes that affected the performance of the pipeline.

TABLE III
GRASP POSE SELECTION RESULTS

	Cylinders					Boxes					Others	
	CC	C	SP	BPS	BPL	IT1	IT2	IT3	WB	BB	DT	B
Success rate (%)	93.3	100	100	100	80	83.3	100	86.7	93.3	83.3	6.7	60



(a) 5 centimeters (b) 2 centimeters (c) 1 centimeter

Fig. 13. Octomap resolution

Pipeline integration results: To test the complete pipeline efficiency, a household possible scenario was created, using a table and putting a cereal box on the table to serve as an obstacle. Then, putting the objects mentioned above on the table on a random pose the grasping action was tested for each of them through ten attempts (see Figure 14 for an example of the coke experiment). The results for the several objects are summed up in table IV.

TABLE IV
FINAL RESULTS

	Cylinders					Boxes					Others	
	CC	C	SP	BPS	BPL	IT1	IT2	IT3	WB	BB	DT	B
Success rate (%)	80	70	70	80	70	60	60	70	70	60	-	0(20)

To sum up, these results show a good success rate in general for the whole pipeline execution. The unsuccessful attempts were related to an imprecise octomap representation of the world that led to the detection of inexistent collisions, small calibration problems and deficient grasp poses, result of noise in the point clouds, that along with the lack of rigidity in the fingers of the gripper led to failed grasps, specially for the bowl case. However, after replacing the bowl by a lighter plastic tupperware with identical geometry, the success rate rose from 0% to 20%.

VI. CONCLUSIONS

This work proposed a methodology to plan grasps in uncontrolled domestic environments using a mobile robot with an arm with seven degrees of freedom. This planning was implemented in a pipeline that is composed of two major parts: perception and motion planning.

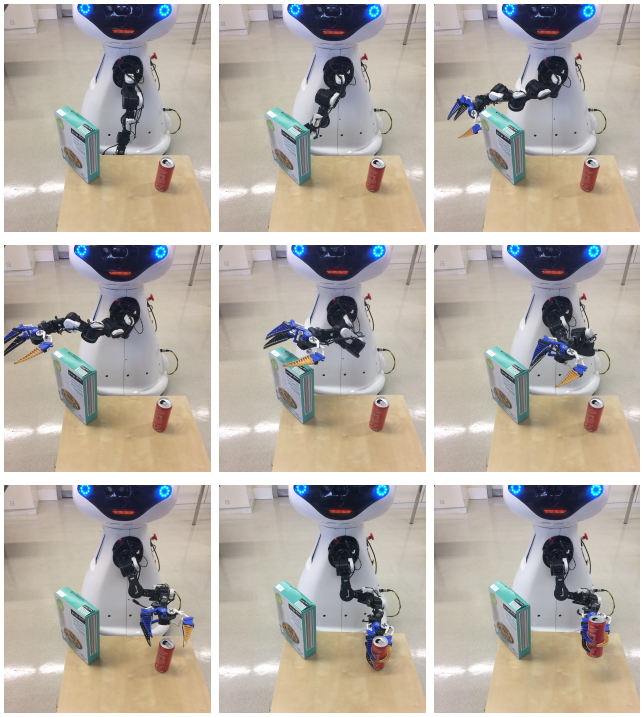


Fig. 14. Trajectory in the real world

The perception was performed by a depth camera mounted at the top of the robot. The environment was perceived and in the object to be grasped was carried out a force-closure antipodal grasp pose detection to select the region where the object should be grasped depending on its geometry and on the robot position. Several objects were used to test this method, which achieved higher success rates for cylinder-shaped objects, which was expected. The implemented method achieved good results in general, specially bearing in mind that it does not depend on the class of the object nor how it was previously known to the robot system, which allows to generalize very well to unknown objects. This was one of the major goals for this work. The motion planning was achieved by implementing the MoveIt! framework to the Cyton Gamma 1500 arm used. In the literature, very few experiments can be found regarding this arm and motion planning with a specific purpose like grasping. As a contribution, this work integrated an additional tolerance algorithm not available in MoveIt! to iterate over a list of poses with an arbitrary deviation from the goal pose to improve the solving rate of the inverse kinematics problem without jeopardizing the grasp execution. Resorting to sampling-based planners, the trajectory of the arm to the goal pose was planned in accordance with the occupancy map to avoid obstacles. As far as the implementation is concerned, the Robot Operating System played a very important role. Its architecture allowed to build an intuitive complete pipeline that performs all the grasping actions, which was ultimately the goal of this work. As future work, it is suggested: the reduction of CPU usage by the octomap in order to increase its resolution and precision, the addition of a DOF on the robot body to allow it to extend vertically and the creation of a feedback method based on the effort applied to the gripper.

REFERENCES

[1] C. C. Kemp, A. Edsinger, and E. Torres-Jara, "Challenges for robot manipulation in human environments," *IEEE Robotics & Automation Magazine*, 2007.

[2] J. R. Napier, "The prehensile movements of the human hand," *The Journal of bone and joint surgery. British volume*, 1956.

[3] T. Feix, J. Romero, H.-B. Schmiedmayer, A. M. Dollar, and D. Kragic, "The grasp taxonomy of human grasp types," *IEEE Transactions on Human-Machine Systems*, 2016.

[4] F. Reuleaux, *The kinematics of machinery: outlines of a theory of machines*. Courier Corporation, 1963.

[5] B. Dizioğlu and K. Lakshiminarayana, "Mechanics of form closure," *Acta mechanica*, vol. 52, no. 1-2, pp. 107–118, 1984.

[6] A. Bicchi and V. Kumar, "Robotic grasping and contact: A review," in *ICRA*, vol. 348. Citeseer, 2000, p. 353.

[7] I.-M. Chen and J. W. Burdick, "Finding antipodal point grasps on irregularly shaped objects," *IEEE transactions on Robotics and Automation*, vol. 9, no. 4, pp. 507–512, 1993.

[8] A. Saxena, J. Driemeyer, and A. Y. Ng, "Robotic grasping of novel objects using vision," *The International Journal of Robotics Research*, 2008.

[9] E. Klingbeil, D. Rao, B. Carpenter, V. Ganapathi, A. Y. Ng, and O. Khatib, "Grasping with application to an autonomous checkout robot," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 2011.

[10] D. Fischinger, A. Weiss, and M. Vincze, "Learning grasps with topographic features," *The International Journal of Robotics Research*, 2015.

[11] L. Montesano and M. Lopes, "Learning grasping affordances from local visual descriptors," in *Development and Learning, 2009. ICDL 2009. IEEE 8th International Conference on*, 2009.

[12] L. Pinto, J. Davidson, and A. Gupta, "Supervision via competition: Robot adversaries for learning tasks," in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, 2017.

[13] S. Levine, P. Pastor, A. Krizhevsky, J. Ibarz, and D. Quillen, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *The International Journal of Robotics Research*, 2018.

[14] E. G. Gilbert, D. W. Johnson, and S. S. Keerthi, "A fast procedure for computing the distance between complex objects in three-dimensional space," *IEEE Journal on Robotics and Automation*, 1988.

[15] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: An efficient probabilistic 3d mapping framework based on octrees," *Autonomous Robots*, 2013.

[16] S. Kumar, N. Sukavanam, and R. Balasubramanian, "An optimization approach to solve the inverse kinematics of redundant manipulator," *International Journal of Information and System Sciences (Institute for Scientific Computing and Information)*, 2010.

[17] H. M. Choset, S. Hutchinson, K. M. Lynch, G. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of robot motion: theory, algorithms, and implementation*. MIT press, 2005.

[18] M. Gualtieri, A. ten Pas, and R. Platt, "Pick and place without geometric object models," in *IEEE Intl Conf. on Robotics and Automation*, 2018.

[19] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, 1998.

[20] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the 22nd ACM international conference on Multimedia*, 2014.

[21] A. Singh, J. Sha, K. S. Narayan, T. Achim, and P. Abbeel, "Bigbird: A large-scale 3d database of object instances," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 2014.

[22] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[23] L.-C. Wang and C.-C. Chen, "A combined optimization method for solving the inverse kinematics problems of mechanical manipulators," *IEEE Transactions on Robotics and Automation*, 1991.

[24] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation*, 2004.

[25] P. Beeson and B. Ames, "Trac-ik: An open-source library for improved solving of generic inverse kinematics." IEEE, 2015.

[26] J. Sequeira, P. Lima, A. Saffiotti, V. Gonzalez-Pacheco, and M. A. Salichs, "Monarch: Multi-robot cognitive systems operating in hospitals," in *ICRA 2013 workshop on many robot systems*, 2013.