Plane-based Humanoid Robot Navigation and Object Model Construction for Grasping

Pavel Gritsenko[2], Igor Gritsenko[2], Askar Seidakhmet[2], and Bogdan Kwolek[1]

[1] AGH University of Science and Technology, 30 Mickiewicza, 30-059 Krakow, Poland
http://home.agh.edu.pl/~bkw/contact.html
[2] Al-Farabi Kazakh National University, Prospect al-Farabi 71, Almaty, Kazakhstan

Abstract. In this work we present an approach to humanoid robot navigation and object model construction for grasping using only RGB-D data from an onboard depth sensor. A plane-based representation is used to provide a high-level model of the workspace, to estimate both the global robot pose and pose with respect to the object, and to determine the object pose as well as its dimensions. A visual feedback is used to achieve the desired robot pose for grasping. In the pre–grasping pose the robot determines the object pose as well as its dimensions. In such a local grasping approach, a simulator with our high-level scene representation and a virtual camera is used to fine-tune the motion controllers as well as to simulate and validate the process of grasping. We present experimental results that were obtained in simulations with virtual camera and robot as well as with real humanoid robot equipped with RGB-D camera, which performed object grasping in low-texture layouts.

Keywords: Object grasping; humanoid robot; pose recovery

1 Introduction

Humanoid robotics technology has recently made rapid progress. However, object grasping by humanoid robots is still a challenging problem due to complex mechanical structures, long kinematic chains, occlusions and noisy camera observations. Conventional approaches to object grasping can fail when the shape, dimension or pose of the objects are missing or are not precise enough. Grasping of unknown objects with neither 3D models nor appearance data is very important for humanoid robots that by definition should work in unknown and unstructured environments. Moreover, any humanoid robot should automatically determine best grasping pose, i.e. be able to approach the object as well avoid obstacles while moving towards the object of interest, and finally take the best pose with respect to the object. As noticed in a recent survey [1], very few works employ visual feedback during the reaching and grasping. Most approaches rely on open-loop algorithms, where the robot takes a single shot of the scene, determines the relative poses between the hands and the object of interest, and finally without a visual feedback drives the arms towards the object.

In real grasping scenarios with humanoid robots, particularly with robots that act as human assistants, the robot should be able to detect the object on the

basis of a description provided by the user, or alternatively, it should recognize the object of interest or acquire information about the object on the basis of pointing gestures. Given the object of interest determined with such a user-friendly interface, the humanoid robot should approach the object and take the pre-grasping pose. Finally, it should determine object dimensions, best grasping points, calculate arm motions and execute grasping task. While approaching the object the robot should, among others, simultaneously build or update the map of the environment, construct high level-representation of the environment, determine its own pose in the map and with respect to the object, detect and avoid obstacles, etc.

Navigating in unknown environment towards the object to be grasped requires a SLAM (Simultaneous Localization and Mapping) system. Several approaches to solve SLAM problem have been proposed in the last two decades [2]. Most existing SLAM systems use RGB cameras or RGB-D cameras to perceive the scene. Traditional RGB image-based SLAM might fail in challenging low-texture cases. The reason for this is that in low-texture scenes it is often difficult to find a sufficient number of reliable point features and, as a consequence, the performance of such SLAM algorithms degrades. In such low-texture scenarios, in addition to points, lines, planes and objects should be utilized. Moreover, reasoning about 3D objects and layouts should be carried out for better scene understanding. The RGB-D SLAM systems built on point clouds require substantial amount of memory. Even when mapping a simple scene, for instance an empty room, in mentioned above representation the memory demands grow quickly with scene complexity and time.

In general, the existing approaches for unknown object grasping can be divided into two groups: global and local grasping approaches. Global grasping approaches employ the full 3D model of the unknown object to find appropriate grasps. The model can be constructed by the use of multiple views of the object, for instance, by using structure from motion to refine existing model, by using point clouds to infer 3D structure and/or model, or by decomposition the object into 3D shapes, etc. In contrast, local grasping approaches only utilize available data to accomplish suitable grasps using information like edges, silhouettes, boundaries, etc.

In this work we present an approach to humanoid robot navigation and object model construction for grasping using only RGB-D data from an onboard depth sensor. A plane-based representation is used to provide a high-level model of the workspace, i.e. the world in which the robot performs the task and occupies space, to estimate both the global robot pose and pose with respect to the object, and to determine the object pose as well as its dimensions. A visual feedback is used to achieve the desired robot pose for grasping. In the first stage, error between robot position and desired path is calculated, whereas in the last stage, error between robot pose and object pose is calculated for visual controller, which allows the robot reaching a pre-grasping pose. In the pre–grasping pose the robot determines the object pose as well as its dimensions. In such a local grasping approach, a simulator with our high-level scene representation and a

virtual camera is used to fine-tune the motion controllers as well as to simulate and validate the process of grasping. We present experimental results that were obtained in simulations with virtual camera and robot as well as with real humanoid robot equipped with RGB-D camera, which performed object grasping in low-texture layouts.

2 Relevant Work and Our Contribution

2.1 Relevant Work

Contrary to analytic approaches to object grasping, approaches following the data-driven paradigm focus on object representation and extraction of perceptual information [1]. Data-driven grasp synthesis started to be used more broadly with the availability of GraspIt! [3], which is an interactive simulation, planning, analysis, and visualization tool to determine stable grasp for a mechanical hand. In [4], Kragic et al. presented a visual tracking system that was capable of recognizing objects. Once an object was recognized the model and object pose have been sent to GraspIt!. Then, a human operator utilized the discussed tool to design motion strategies for grasping of the considered object. The latest version of the discussed simulator can accommodate arbitrary hand as well as load objects and obstacles of arbitrary geometry.

In grasping system of a humanoid robot we can distinguish units responsible for navigation of the robot, acquisition of the object model and determining grasping points from the acquired data. Usually, the navigation unit is considered separately or it is assumed that the robot is placed in vicinity of the object, i.e. it is already located in a pre-grasping position. In [5] a SLAM-based grasping framework for robotic arm navigation and object model construction is presented. Authors of discussed work argue that although improving the object model is beneficial, it is not enough to achieve stable grasping. Thus, a correction of robot pose has been one of the key aspect of the proposed framework. They demonstrated the influence of the accuracy of the robot pose on the process of grasping. A simple object constructed from three planes attached together has been used in all experiments. The features of the object were defined as the intersection between these planes.

In approaches belonging to local grasping category, RGB-D cameras are usually used to provide information required for completing the task. An approach [6] uses object edges to determine grasping locations by fitting a so-called grasping rectangle on image plane. Such a rectangle is used to describe gripper configuration. A learning algorithm has been investigated to select the best grasping location depending on the object shape. In a later work [7], the presented above idea has been extended by looking at the contact area of the grasping rectangle. For instance, if the contact area appears to be too small, the grasp task is likely to fail and a better grasp is searched. An approach proposed by Ala et al. [8] retrieves graspable boundaries and convex segments of unknown object. In point clouds acquired by a 3D camera a scene segmentation is conducted to determine

point cloud belonging to the object. With the help of blob detection, the object boundaries are determined to extract the boundary edges. The grasp planner determines one contact point in order to execute a boundary grasp. In [9], the algorithm fits the gripper shape to point cloud that belongs to the object. It utilizes a segmentation to delineate object from the scene as well as incorporates learning to improve the grasp success rate. In approach of Navarro [10], object center is estimated on the basis of point cloud. A tracking algorithm has been used to extract objects on a conveyer belt and then grasp them by a gripper. In [11] the principal axis and centroid of unknown object are estimated on the basis of point cloud to determine a stable grasp. A high success rate has been demonstrated for a set of household objects.

On the basis of a bottom-up hierarchical clustering approach, which is able to segment objects and parts in a scene, a transform from such a segmentation into a corresponding, hierarchical saliency function has been proposed in [12]. This hierarchical saliency characterizes most salient corresponding region (scale) for each point in the image. The discussed algorithm has been evaluated on an easy-to-use pick and place manipulation system.

2.2 Differences with Relevant Approaches

Our approach to scene and object representation differs in several aspects from approaches discussed in relevant work. The first difference is that our algorithm does not decompose point clouds onto sets of separate primitives, but instead it aggregates detected in advance planes onto objects and builds complex representations comprising plane-based objects and point clouds. The second difference is that our algorithm reconstructs any type plane-based objects consisting of any number of faces. Thus, it differs from widely used RANSAC algorithm, which can extract objects described mathematically. It can employ any of the state-of-the-art closed-form solutions [13,14,15,16,17,18] to estimate the robot pose. The use of trihedral angles (corners) to represent plane-based objects permits the discussed capability. Trihedral angles permit the use of all three types of primitives: point, line, plane, as well as point-to-point [15], line-to-line [14], plane-to-plane [13], point-to-line [17], point-to-plane [18], and line-to-plane [16] correspondences and their closed-form solutions. Owing to use of graph patterns it can be used to perform classification of objects.

Another important point is that there exist no state-of-the-art representation providing high-level information about the environment to allow a robot to distinguish between objects, their sizes and types [19]. With the ability to extract objects and their physical sizes, our high-level map representation provides additional information and opportunities for higher level understanding of the scene and object geometries.

3 Algorithm overview

The plane object-based map representation is built on point cloud, which is calculated on the basis of RGB-D streams delivered by the Xtion sensor. Upon the segmented planes in the point cloud we build a graph, where a node and edge represent a plane and its real intersection with other plane, respectively. Afterwards, we extract all trihedral angles, i.e. corners represented by 3rd order cycles in the graph. Next, we carry out systematic aggregation of trihedral angles onto object such as trihedral angles belonging to the same plane-based object have common edges. At the end, we perform object classification using simple subgraph patterns and calculate their physical sizes. During approaching the object by a humanoid robot with the onboard RGB-D sensor, the point could is segmented, planes are extracted, correspondences between planes are determined, and relative poses (rotations and translations) between current and previous point clouds are determined using RANSAC algorithm, which is initialized in each frame with a direct pose estimate [20]. On the basis of the initial pose of the robot in real world coordinates, which is determined automatically on the basis of visual data, as well as object position that can be determined automatically or specified by the user, the robot calculates a collision free path, and then follows it using a visual feedback. A controller responsible for path following and reaching the desired pose uses the robot pose as well as object location. In the vicinity of the object the controller employs the object pose to achieve best grasping pose. It is determined by a simulator, which on the basis of our high-level object representation, the kinematic model of the humanoid robot, as well as virtual camera, performs a simulation of reaching and grasping the object. This means that the 3D scene fragments that are located in the field of view of the virtual camera can be extracted and then projected into the image plane of the camera. The simulations were done using Webots robot simulator [21,22], which provides a complete development environment to model, program and simulate robots and our scripts in Python language. In the simulation experiments an ArUco marker [23], which was attached to the object of interest has been detected and then used to determine the pose of Nao robot acting in virtual world. The parameters obtained in the simulation were then used in real experiments. Given a motion controller tuned by the discussed tool, the robot can follow any feasible path towards the pre-grasping position. The main role of our tool is simulation and visualization of the grasping process given the real pose of the robot as well as pose of the object together with its dimensions. We assume that the objects undergoing grasping are located on the floor and are composed of planes.

Figure 1 depicts step–by–step transition from simple point cloud to classified plane–based object representation with sizes (edge lengths). In the algorithm we can distinguish seven major steps. In the first step, RGB-D data stream consisting of a pair of a color image and a depth map is acquired. After that, colored point cloud is reconstructed on the basis of the RGB-D stream and intrinsic calibration parameters. The colored point cloud is an initial representation of the observed scene. In the next step, M-estimator SAmple Consensus (MSAC)

is executed to perform plane segmentation. Afterwards, real planes intersections with their lengths are calculated. At the end, on the basis of segmented planes and their intersections a graph is being built. This leads to second map representation that comprises plane equations and point clouds representing objects of complex shape (not plane-based objects). Due to substitution of point clouds representing plane equations the map representation gets compact. In the following step, extraction of trihedral angles that are represented by 3rd order cycles from the constructed graph takes place (Fig. 2). After determining the trihedral angles, their aggregation can be performed as the trihedral angles of the same plane-based object have common edges. Finding common edges for trihedral angles is performed using the constructed graph. Through aggregating all trihedral angles with common edges we get abstract plane-based objects. This means that at this stage we have a structure that describes which planes belong to which object, the number of such objects, their dimensions, but no types. Therefore, this representation is called as abstract plane-based objects, see also Fig. 1. The last step is devoted to classification. The classification relies on matching graph patterns with a subgraph from the extracted graph. The resulting algorithm is simple and effective.

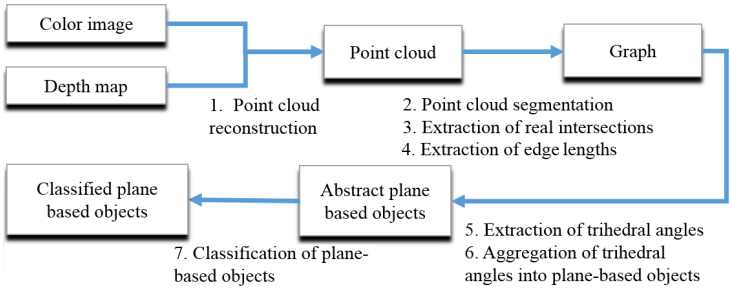

Fig. 1. Diagram illustrating transition from colored point cloud to classified plane object-based high-level map representation.

4 The Algorithm

4.1 Object model and its dimensions

Determining physical object size. Let us consider how the real intersection of two planes and their length can be extracted. Let's assume that we have two planes: $P_1 = A_1x+B_1y+C_1z+D_1$ and $P_2 = A_2x+B_2y+C_2z+D_2$. According to algorithm in Fig. 1, after point cloud segmentation we determine the plane intersections. Thus, after this step we have in disposal planes and their inliers, i.e. array of points representing them. Let's denote P_1 inliers as $\{p_{plane1,1} \dots p_{plane1,i}\}$ and P_2 inliers as $\{p_{plane2,1} \dots p_{plane2,i}\}$. According to definition of the intersection, which states that if planes P_1 and P_2 have real intersection then they should

have common inliers $\{p_{edge,1} \dots p_{edge,i}\}$, i.e. array of points belonging to P_1 and P_2 simultaneously, see (1):

$$\{p_{edge,1} \dots p_{edge,i}\} = \{p_{plane1,1} \dots p_{plane1,i}\} \cap \{p_{plane2,1} \dots p_{plane2,i}\} \quad (1)$$

However, $p_{edge,1} \dots p_{edge,i} = \emptyset$, because a point can be among inliers of only single plane. Thus, we reformulate this definition and state that the points representing real intersection $p_{edge,1} \dots p_{edge,i}$ will be simultaneously close to planes P_1 and P_2. Therefore, on the basis of equations (2)–(5) we determine the point-to-plane distance for both sets of inliers to both planes.

$$\{d_{plane1-plane1,1} \dots d_{plane1-plane1,i}\} = |A_1 p_{plane1,i,x} + B_1 p_{plane1,i,y} + C_1 p_{plane1,i,z} + D_1| * (A_1^2 + B_1^2 + C_1^2)^{1/2} \quad (2)$$

$$\{d_{plane1-plane2,1} \dots d_{plane1-plane2,i}\} = |A_2 p_{plane1,i,x} + B_2 p_{plane1,i,y} + C_2 p_{plane1,i,z} + D_2| * (A_2^2 + B_2^2 + C_2^2)^{1/2} \quad (3)$$

$$\{d_{plane2-plane2,1} \dots d_{plane2-plane2,i}\} = |A_2 p_{plane2,i,x} + B_2 p_{plane2,i,y} + C_2 p_{plane2,i,z} + D_2| * (A_2^2 + B_2^2 + C_2^2)^{1/2} \quad (4)$$

$$\{d_{plane2-plane1,1} \dots d_{plane2-plane1,i}\} = |A_1 p_{plane2,i,x} + B_1 p_{plane2,i,y} + C_1 p_{plane2,i,z} + D_1| * (A_1^2 + B_1^2 + C_1^2)^{1/2} \quad (5)$$

After that we determine all points that have the distance to both planes smaller than d_{thresh} and grater than d_{min}. d_{min} is utilized to cope with noise.

In order to define edge length d_{edge} we take two points from $p_{edge,1} \dots p_{edge,i}$ with maximum distance to each other. The distance between the points represents diagonal length $d_{diagonal}$ of the cylinder of radius d_{thresh}. Thus, the edge length is determined on the basis of (6):

$$4 * d_{tresh}^2 + d_{edge}^2 = d_{diagonal}^2 \quad (6)$$

Aggregation of planes into object. After determining plane intersections we can build a graph whose node and edge represent a plane and its real intersection with other plane, respectively. The graph permits us to determine in a fast manner all trihedral angles as 3rd order cycles in the graph, see also Fig. 2. Subsequently, we assume that all trihedral angles of the same plane-based object have common edges.

If the object is presented by several trihedral angles the assumption mentioned above allows us to merge them angle-by-angle and extract plane-based object. If the trihedral angle has no common edges then the object is represented by single trihedral angle. All trihedral angles are represented by three planes, see also Fig. 2. Let's denote them by indexes of planes. Thus, the trihedral angle 1-4-7 has no common edges. Therefore, Box object from Fig. 2 is plane-based object and is represented by single trihedral angle, whereas Locker 2 object is

represented by single trihedral angle 1-3-6. Locker 1 object from Fig. 2 consist of two trihedral angles 1-8-9 and 1-5-8. These trihedral angles have common edge 1-8, so that Locker 1 is plane-based object represented by two trihedral angles. In this way the aggregation is capable of adding trihedral angles, angle-by-angle.

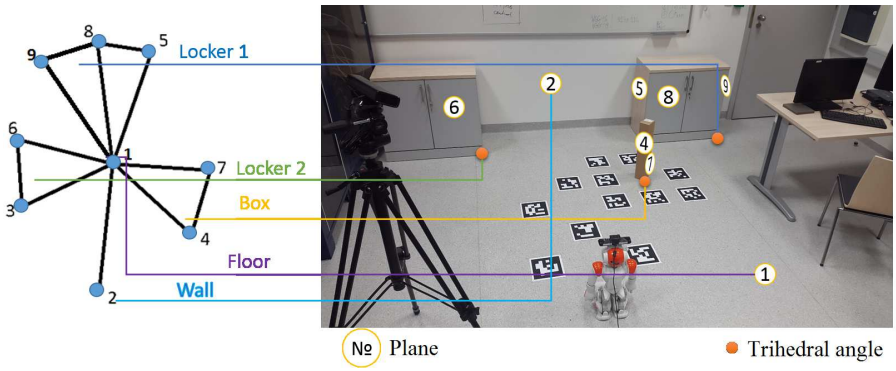

Fig. 2. Graph representing scene that is observed by Nao robot with onboard Xtion sensor. Orange stands for centers of trihedral angles. Circles with numbers stand for segmented planes, whereas blue color points on the graph vertices. Black edges between blue nodes on the graph represent real intersections between segmented planes. For visualization purposes only selected trihedral angles were used for graph construction.

Classification of objects. After extracting all plane-based objects, a classification step is executed. This step boils down to splitting the graph into sub-graphs (for instance plane one in Fig. 2), and then matching the predefined patterns, see Fig. 3.

Fig. 3. Predefined patterns to match objects in the graph representation.

Merging plane object-based high-level map representation. Merging a pair of plane object-based high-level map representation is performed using a graph representation. At the beginning, data association is performed, and plane-to-plane correspondence is determined on the basis of matching normals and distances between the considered planes. Afterwards, a transformation matrix

between frames is determined on the basis of closed-form solution for plane-to-plane correspondence [20]. The transformation matrix is then refined by the ICP algorithm. Next, frames are aligned. Subsequently, on the basis of plane-to-plane correspondences we merge graph representations. This is done under assumption that each edge length is the longest one from corresponding edges in pair of frames. In this way the graph is reconstructed. In the last stage, we reinitialize the plane-based object structure and execute classification, see Fig. 4.

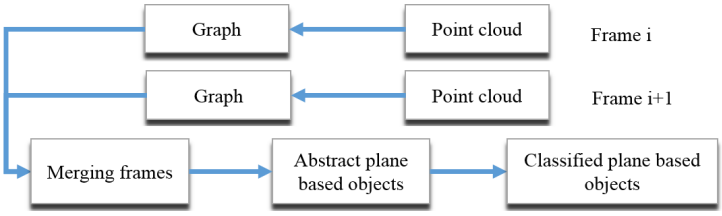

Fig. 4. Algorithm of merging plane object-based high-level map representations.

Determining the main axis of the object. Let's denote center of trihedral angle as $T_c(x_{T_c}, y_{T_c}, z_{T_c})$, and the three edge vectors containing edge length and its direction as $(\boldsymbol{e_1}, \boldsymbol{e_2}, \boldsymbol{e_3})$. Then central point of the object $C(x_c, y_c, z_c)$ can be calculated as follows:

$$C = T_c + (\boldsymbol{e_1} + \boldsymbol{e_2} + \boldsymbol{e_3})/2 \tag{7}$$

If object includes several trihedral angles then central point of the object is identified as mean of central points of all trihedral angles in this object. The axis is defined as normal $\boldsymbol{N}(A, B, C)$ of the plane described by the equation $P = Ax + By + Cz + D$ and representing object side chosen for grasping. From one hand, the proposed algorithm can identify central point and appropriate sides for grasping, as well as their axes for any plane-based object. From the other hand, complex plane-based objects require complex control strategy for grasping and it is an open problem. Figure 5 illustrates identification of sides and axes of the object, which are appropriate for grasping, as well as calculating object's central point.

4.2 Navigation and Grasping

Given the point cloud, the robot extracts planes from it and identifies correspondences between planes. Then, on the basis of such plane correspondences, not less than three, not parallel planes are used for direct 6-DoF pose estimation [20]. This pose is used as an initial point for further refinement by the ICP algorithm. It is worth noting that the ICP algorithm uses no initial point cloud, but it uses reconstructed point cloud on the basis of inliers of all extracted planes. Such point cloud is constructed for the current frame and for the target

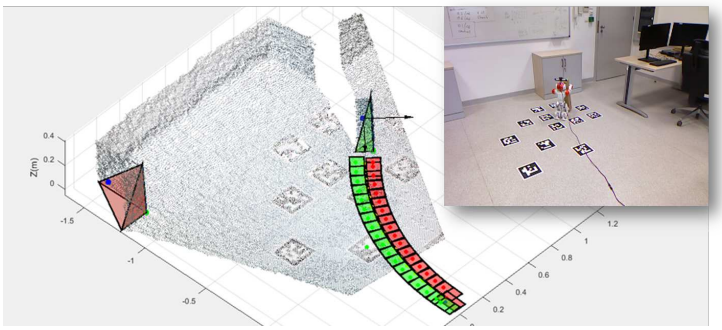

Fig. 5. Identification of sides and axes of the object, which are appropriate for grasping, as well as calculating object central point. Motion planning was done on the basis of extracted central point and axis of proper side. Red patches represent footsteps of right leg, while green patches represent footsteps of left leg. Different objects on the scene are represented by different colors, e.g. Box from Fig. 2 is represented by green color, while Locker 2 is represented by red color. No axis is presented for Locker 2 because no side is appropriate for the grasping.

frame to align. After alignment, not associated plane equations are added to the first frame together with their inliers. The transformation for the next frame is calculated with regard to the updated first frame. Therefore, transformation is calculated not in frame-by-frame manner, but all frames are aligned to the updated first frame. As a result of the alignment, final transformation matrix and robot pose are determined. To cope with data association we keep transformation matrix for the last used frame, so that each new frame is firstly transformed by this matrix.

5 Experimental Results

Our system is conceived to work with any RGB-D sensor. In the experiments we utilized ASUS Xtion sensor, which has been mounted on the Nao humanoid robot. It works at a frame rate of 30 Hz and delivers stream of color images of size 320×240 and depth map stream, whose images have resolution of 640×480 pixels. The system has been tested in scenarios similar to scenario shown on Fig. 2. Figure 6 illustrates constructing plane object-based high-level map representation and merging it frame-by-frame. Figure 6a shows results of reconstruction of point cloud using color image, depth map and intrinsic calibration parameters. Figure 6b depicts extraction of planes from the reconstructed point cloud using MSAC. Figure 6c shows results of systematic aggregation of trihedral angles into objects. The top row presents results that were obtained for the first frame, whereas bottom row presents results that were obtained in frame tenth. In the discussed experiment we merged point clouds that were reconstructed step-by-step from color images and depth maps, using estimated robot poses.

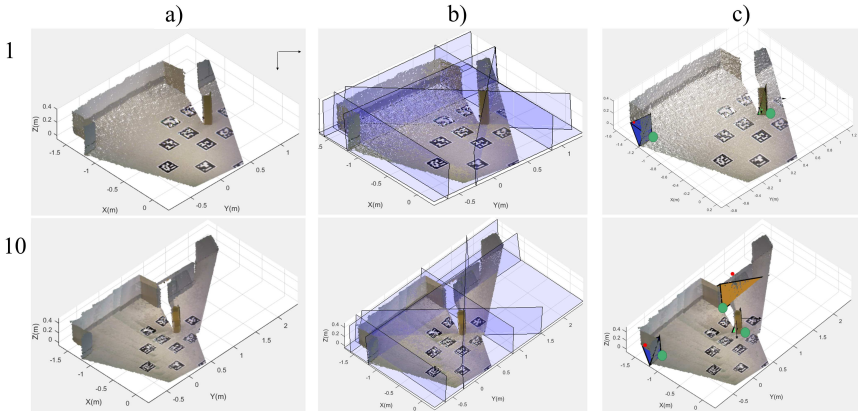

Fig. 6. An example of constructing plane object-based high-level map representation and merging it frame-by-frame. (a) Reconstructed point cloud using color image, depth map and intrinsic calibration data of the sensor. (b) Extracted planes from point cloud using MSAC. (c) Systematic aggregation of trihedral angles into objects. Green points of the graph denote centers of trihedral angles, while blue points denote estimated centers of objects. Edges of trihedral angles are represented by black line segments, while axis of objects appropriate for grasping are represented by black arrows.

As we can observe on Fig. 6b–c, the proposed algorithm for extraction of real intersections of planes from point cloud gave satisfactory results. It is worth noting that the trihedral angle permits the use of all three types of basic geometric primitives: point, line and plane. However, how to combine transformation matrices, which are calculated using different closed form solutions for our high-level map representation is an open problem. In the discussed experiment we have utilized plane-to-plane [13] correspondence in order to find the transformation matrix between frames.

In the next stage we conducted experiments consisting in grasping the objects by Nao robot. At the beginning we performed simulations to tune controllers for path following. Afterwards, simulations aiming at tuning arm controllers were conducted. Finally, Nao humanoid robot has been utilized in experiments with object grasping. Figure 7a shows view of the simulated scene using Webots API (see also images acquired by virtual cameras), Fig. 7b depicts view of the real scene, whereas Fig. 7c contains example images that were obtained in simulation of object grasping. In Figure 7a there are three sub-images that were acquired by virtual cameras that the Nao robot was equipped with. The left sub-image was acquired by camera (bottom) that is located in the Nao's head, the middle sub-image contains depth map that was grabbed by simulated RGB-D sensor, whereas right sub-image depicts RGB image acquired by RGB-D sensor. As we can notice, the RGB-D sensor has been mounted on the Nao's head.

In order to show potential of the presented approach we calculated errors in three scenarios. In the first one the Nao robot approached the object without

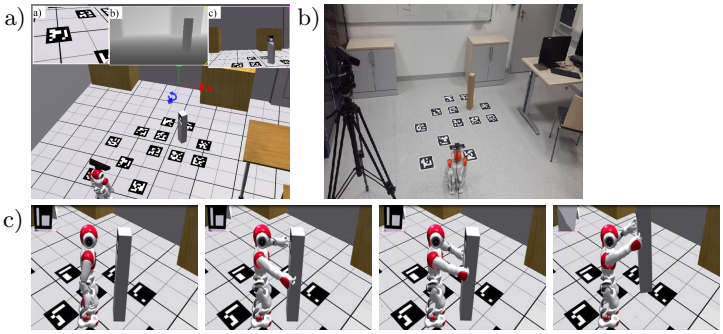

Fig. 7. View of the simulated scene a), real scene view b), simulated grasping of the object c).

visual feedback using only footsteps determined in advance. In the second scenario the robot followed path that was determined in advance. In the discussed scenario the robot pose has been corrected using visual feedback in a predefined number of control points. In the last scenario the robot followed first part of the path using visual feedback in the predefined number of control points, whereas in the last part of the path the information about the relative robot-object poses was used to correct the robot motion. The actual robot pose in the world coordinates has been determined on the basis of images acquired from on-board camera, using algorithm discussed in Subsection 4.2. The pose and main axis of the object were determined using algorithm described in Subsection 4.1. Figure 8 depicts representative results that were obtained in one of the simulations.

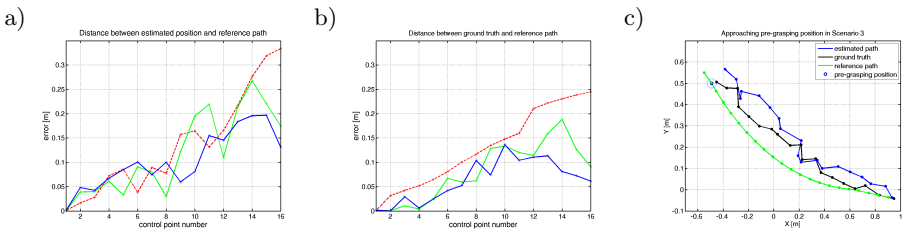

Fig. 8. Left plot: distance between estimated position and the reference path vs. control point number. Middle plot: distance between ground-truth and the reference path. Red - no correction of Nao pose, green - path following using visual feedback, blue – path following in the first eight control points and object approaching in remaining points. Right plot: approaching pre-grasping position in Scenario 3.

Figure 8a depicts distance between estimated position and the reference path vs. control point number in three scenarios. In the first scenario the robot walked

towards the object using motions that were determined in advance by the footstep planner. As we can notice, without a visual feedback the error grows over time. Owing to using P controllers that were responsible by steering the robot towards the reference path, the error between the reference path and the estimated pose has been reduced. The control corrections have been calculated using the estimated robot position and orientation with respect to reference position and orientation and then used in moveTo function to correct robot motion. In the discussed scenario, in each control point the robot corrects its motion to follow the reference path using visual feedback. In the third scenario, the robot followed first part of the path using the same control strategy as in scenario two, whereas in the last part of the path the robot used information about the object location. This means that in the control point the robot estimated relative position and orientation with respect to the object of interest. On the basis of estimated object position and orientation the robot exploited P controllers to reduce distance to the object and angle between camera main axis and object main axis. As we can notice on the plot shown on Fig. 8a, in the discussed scenario the error at the pre-grasping position has been further reduced.

Figure 8b depicts distance between the reference path and the ground-truth, i.e. distance between the reference path and the real position of the robot in three scenarios. As we can notice, in the third scenario, in the pre-grasping position the error between the robot and the desired position is equal to 6.6 cm. Table 1 contains errors that were obtained in discussed experiment. As we can see on Fig. 8c the robot was able to achieve a pre-grasping position that was quite close to desired one.

Table 1. Errors that were obtained in the simulated environment, c.f. plot on Fig. 8b.

Error [cm]	Scenario 1	Scenario 2	Scenario 3
Average error	13.0	8.3	6.6
Error in pre-grasping position	24.5	9.1	6.1

Figure 9 depicts some color images with corresponding depth maps that were acquired by virtual RGB-D camera mounted on Nao's head in an experiment in 3rd scenario. The advantage of our simulator is that it not only provides precise ground-truth for both the robot and the camera but it allows also contamination of images and depth maps by noise that can arise during real experiments. This means that it allows to investigate the influence of blur motion, rapid movements and rotations as well as depth error on the performance of the object grasping.

Next, we conducted experiments using real humanoid robot. The ground-truth of the robot pose has been determined using ArUco markers [23], which were placed on the floor, see also Fig. 2. In 3rd scenario the error at the pre-grasping position was between five and eight centimeters. The average error in determining the physical dimensions of the rectangular object from Fig. 2 is

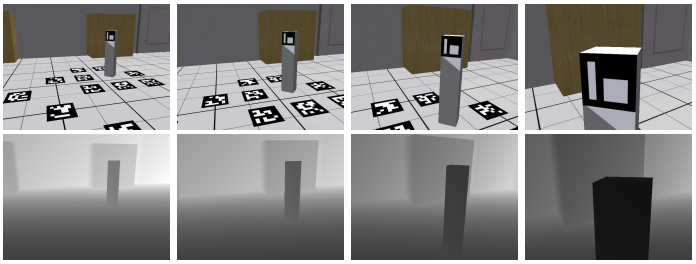

Fig. 9. Images acquired by virtual RGB-D camera mounted on Nao's head in control point #1, #6, #11 and #16, respectively.

about two centimeters, whereas the average error of the main axis is below three degrees. Figure 10 depicts sample images that were acquired in one of the experiments. As we can notice, due to rapid rotations the images are contaminated by motion blur, and thus the ground-truth that has been calculated on the basis of ArUco markers was not accurate enough to calculate supplementary quantitative results. The system operates at frame rate of about 5 Hz at Intel I7 CPU. Supplemental material from experiments with real robot is available at: `http://bit.ly/ECCV2018_Wksp_6DObjectPose`.

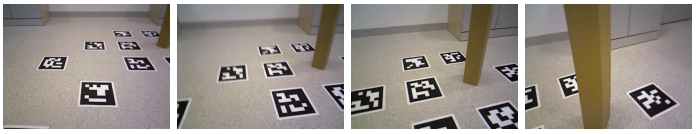

Fig. 10. Images acquired by Xtion sensor mounted on Nao's head in control point #1, #6, #11 and #16, respectively.

6 Conclusions

We presented an approach for plane-based humanoid robot navigation towards the object as well as object model construction for grasping. We presented a novel algorithm for constructing plane object-based high-level scene representation. We discussed determining the object model and its dimensions, aggregation of planes into object, and determining the main axis of the object. We presented experimental results that were achieved in simulations with virtual camera and robot as well as with real humanoid robot equipped with RGB-D camera, which performed object grasping in low-texture layouts. The visual feedback acknowledged its usefulness in achieving the pre-grasping pose by the robot.

Acknowledgment. This work was supported by Polish National Science Center (NCN) under research grants 2014/15/B/ST6/02808 and 2017/27/B/ST6/01743.

References

1. Bohg, J., Morales, A., Asfour, T., Kragic, D.: Data-driven grasp synthesis – A survey. IEEE Trans. on Robotics **30**(2) (2014) 289–309
2. Bresson, G., Alsayed, Z., Yu, L., Glaser, S.: Simultaneous localization and mapping: A survey of current trends in autonomous driving. IEEE Trans. on Intelligent Vehicles **2**(3) (2017) 194–220
3. Miller, A.T., Allen, P.K.: Graspit! A versatile simulator for robotic grasping. IEEE Robotics Automation Magazine **11**(4) (2004) 110–122
4. Kragic, D., Miller, A.T., Allen, P.K.: Real-time tracking meets online grasp planning. In: Proc. of IEEE Int. Conf. on Robotics and Automation. (2001) 2460–2465, vol. 3
5. Wongwilai, N., Niparnan, N., Sudsang, A.: SLAM-based grasping framework for robotic arm navigation and object model construction. In: IEEE Int. Conf. on Cyber Technology in Automation, Control and Intelligent. (2014) 156–161
6. Jiang, Y., Moseson, S., Saxena, A.: Efficient grasping from RGBD images: Learning using a new rectangle representation. In: IEEE Int. Conf. on Robotics and Automation. (2011) 3304–3311
7. Lin, Y.C., Wei, S.T., Fu, L.C.: Grasping unknown objects using depth gradient feature with eye-in-hand RGB-D sensor. In: IEEE Int. Conf. on Automation Science and Engineering (CASE). (2014) 1258–1263
8. Ala, R., Kim, D.H., Shin, S.Y., Kim, C., Park, S.K.: A 3D-grasp synthesis algorithm to grasp unknown objects based on graspable boundary and convex segments. Inf. Sci. **295**(C) (2015) 91–106
9. ten Pas, A., Platt, R. In: Using Geometry to Detect Grasp Poses in 3D Point Clouds. Springer (2018) 307–324
10. Navarro, S.E., Weiss, D., Stogl, D., Milev, D., Hein, B.: Tracking and grasping of known and unknown objects from a conveyor belt. In: ISR/Robotik 2014; 41st Int. Symp. on Robotics. (2014) 1–8
11. Suzuki, T., Oka, T.: Grasping of unknown objects on a planar surface using a single depth image. In: IEEE Int. Conf. on Advanced Intelligent Mechatronics (AIM). (2016) 572–577
12. Klein, D.A., Illing, B., Gaspers, B., Schulz, D., Cremers, A.B.: Hierarchical salient object detection for assisted grasping. In: IEEE Int. Conf. on Robotics and Automation (ICRA). (2017) 2230–2237
13. Grimson, W., Lozano-Perez, T.: Model-based recognition and localization from sparse range or tactile data. Int. J. of Robotics Research **3**(3) (1984) 3–35
14. Zhang, Z., Faugeras, O.: Determining motion from 3D line segment matches: A comparative study. Image and Vision Computing **9**(1) (1991) 10 – 19
15. Umeyama, S.: Least-squares estimation of transformation parameters between two point patterns. IEEE Trans. Pattern Anal. Mach. Intell. **13**(4) (1991) 376–380
16. Chen, H.: Pose determination from line-to-plane correspondences: Existence condition and closed-form solutions. IEEE Trans. on Pattern Analysis and Machine Intelligence **13**(6) (June 1991) 530–541
17. Nister, D.: A minimal solution to the generalised 3-point pose problem. In: IEEE Conf. Computer Vision and Pattern Recognition (CVPR). (June 2004) 560–567
18. Ramalingam, S., Taguchi, Y.: A theory of minimal 3D point to 3D plane registration and its generalization. Int. J. of Computer Vision **102** (2012) 73–90
19. Cadena, C., Carlene, L., Carrillo, H., Latif, Y., Scaramuzza, D., Neira, J., Reid, I., Leonard, J.: Past, present, and future of simultaneous localization and mapping:

Toward the robust-perception age. IEEE Trans. on Robotics **32**(6) (2016) 1309–1332
20. Khoshelham, K.: Direct 6-DoF pose estimation from point-plane correspondences. In: Int. Conf. on Digital Image Computing: Techniques and Applications (DICTA). (2015) 1–6
21. Michel, O.: Webots: Professional mobile robot simulation. Journal of Advanced Robotics Systems **1**(1) (2004) 39–42
22. Webots: http://www.cyberbotics.com Commercial Mobile Robot Simulation Software.
23. Romero-Ramirez, F.J., Muñoz-Salinas, R., Medina-Carnicer, R.: Speeded up detection of squared fiducial markers. Image and Vision Computing **76** (2018) 38 – 47