

Reuleaux: Robot Base Placement by Reachability Analysis

Abhijit Makhal*
 Department of Mechanical Engineering
 Idaho State University
 Pocatello, Idaho 83209
 Email: makhabhi@isu.edu

Alex K. Goins
 ROS-Industrial
 Email: alex.goins@swri.org

Abstract—Before beginning any robot task, users must position the robot's base, a task that now depends entirely on user intuition. While slight perturbation is tolerable for robots with moveable bases, correcting the problem is imperative for fixed-base robots if some essential task sections are out of reach. For mobile manipulation robots, it is necessary to decide on a specific base position before beginning manipulation tasks.

This paper presents Reuleaux, an open source library for robot reachability analyses and base placement. It reduces the amount of extra repositioning and removes the manual work of identifying potential base locations. Based on the reachability map, base placement locations of a whole robot or only the arm can be efficiently determined. This can be applied to both statically mounted robots, where the position of the robot and workpiece ensure the maximum amount of work performed, and to mobile robots, where the maximum amount of workable area can be reached. The methods were tested on different robots of different specifications and evaluated for tasks in simulation and real world environment. Evaluation results indicate that Reuleaux had significantly improved performance than prior existing methods in terms of time-efficiency and range of applicability.

I. INTRODUCTION

The robotic industry's ongoing advancements depend upon state-of-the-art equipments- with unique specifications and capabilities. As the hardware of the robotic systems are developed, so as the softwares are metamorphosed. From the perspective of an industry or household user, however, the main goal of deploying a robotic system remains the same: to successfully perform the desired task. Tasks can range from intense industrial work such as welding, packaging, and manipulation to relatively smaller scale of picking, placing or grasping. For users to accept and employ a robotic system, it must, (1) precisely execute the task, and (2) integrate well into the user's workspace. For both features, the user and the robot should be aware of information about the robot's reach and the workspace. Without this knowledge, deployment of the robot depends solely on user intuition. An incorrect intuition can lead to human casualty or catastrophic damage to the robotic system and workplace.

Non-expert users who do not have an in-depth understanding of robot kinematics and the reachability challenges of robots, might hold misconceptions that the workspace of a robotic manipulator is sphere-shaped when the radius of the arm is fully extended and in this region the robot's end-effector

can move freely. On the contrary, the robotic arm's workspace depends fully on the constraints posed on its arm joints. In the workspace, there are few positions where the arm can reach freely; in some sections of the workspace, the arm faces singularity. In this paper, the authors conferred few methods by which the robot's workspace could be fully utilized and a suitable placement of the robot or manipulator base can be achieved.

The paper makes the following three contributions to the field of robotic workspace analysis and base placement, 1) A method to generate and analyze the precise reachability of any generic robotic arm in a time-efficient manner, 2) A method to localize the feasible base positions of the robot for any given user task and 3) A comparison and evaluation of these methods on different robots with distinct characteristics.

II. RELATED WORK

One of the challenges in developing a useful robotic system is designing a platform or workspace that lets the robot to effectively complete the desired task. Initial work on path planning and reachability was typically performed for simple grasp points on objects [1], which involves creating a map representing the areas of high dexterity for the manipulators. This work was then extended to the use of reachability maps to solve the inverse reachability task [2] and [3], where the optimal base placement was found in order to perform the desired grasp on an object. Work done by [4] examined ways to simplify the reachability map by generating a capability map, a simplified structure that permits faster and more efficient searches of the map in order to solve the inverse reachability problem.

These methods are limited as they solve only the general problem of whether or not an inverse kinematics solution was found. A given grasp location could mean that some or all of the solutions found could be non-optimal (near singularities or joint limits). The work of [5] presents the concept of manipulability ellipsoid measure, which can be derived from a Jacobian matrix of manipulators. The size of the ellipsoid and the principal axes represent the manipulation ability in a certain configuration and are thus used to determine the effectiveness of a grasp at a given location. This work was extended in terms of grasping and manipulation in [6].

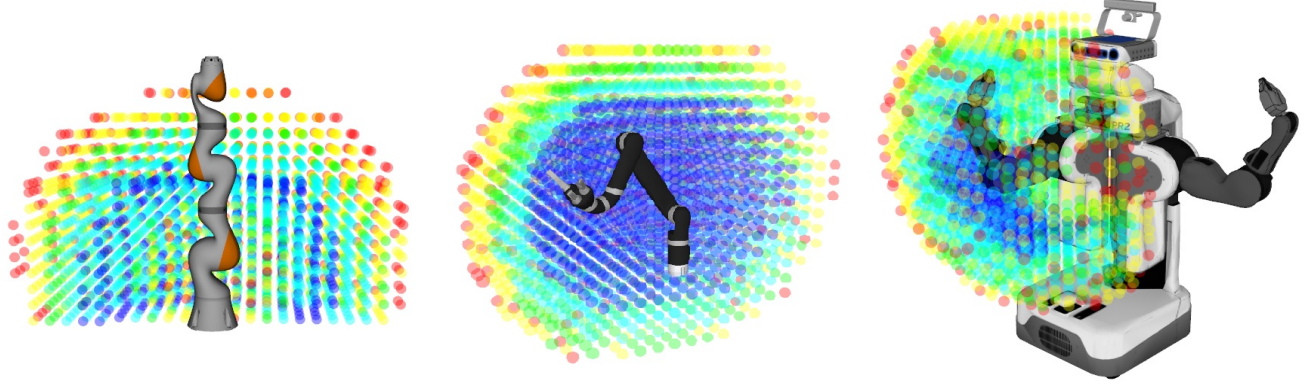


Fig. 1. Reachability Map representation. From left to right a) LWR 7DOF arm with self-collision checking, b) 6DOF JACO arm without self-collision checking, c) right arm (7DOF) of PR2 mobile manipulator with self-collision checking. (Color representation with increasing reachability- Red, Yellow, Green, SkyBlue and Blue)

To overcome the single grasp location issue, several papers extended the inverse reachability problem to solve for trajectories. Various methods have been used to search the reachability map in order to find the location where the desired trajectory can be executed with the highest level of dexterity. [7] uses sampling of the trajectory to find and overlay multiple base placements; [8] uses a pattern search to fit the trajectory into the area representing the field of high dexterity; and [9] uses a cross-correlation technique to fit the desired trajectory to the model of the robot reachability map. Further improvements to the reachability map method were developed by [10] to include the ability to add a transform offset from the original end-effector location, which is useful if the robot is grasping a tool with a non-zero length.

These methods allow planning for simple tasks, but they require a task to be completely specified prior to evaluation. [11] explores the case where a specific trajectory is not given; rather the task has been simplified into a generalized workspace environment where the robot must work. Here, competing constraints are given. However, the operator still evaluates and confirms the final base placement of the robot on the mobile platform. For a simple operation with few operating points, this can be done manually, but for more complex parts and tasks such as welding pipes or assembling parts in constrained spaces, a manual approach for validation may not be feasible.

III. REACHABILITY MAP

A reachability map is a collection of all poses that the robot's end effector can reach. To accommodate the infinite number of reachable poses, similar poses are clustered into structures suitable for visualization and easy to access. The structures also capture the directional information of reachable poses, because several positions can be reached only from a certain direction. An individual part of the structure is represented as multimap data, which holds information about its position, all reachable poses belonging to the structure, and the reachability measure of that structure.

Algorithm 1 Generate Reachability Map

input (URDF of robot)

```

1: procedure GenerateReachMap
2:   create VoxelStructure  $V$ 
3:   for each voxel  $v_i$  in  $V$  do
4:     create sphere  $s$ 
5:     checkforSelf Collision( $s$ )
6:     Store  $s_{filtered}$  in  $S$ 
7:   end for
8:   for each  $s_i$  in  $S$  do
9:     Sample surface and generate poses  $P$ 
10:    for each  $j$  in  $P$  do
11:      findIKSolution( $P_j$ )
12:      if solution then
13:        Store  $(s_i, P_j)$ 
14:      end if
15:       $d_i = FindReachabilityMeasure(s_i)$ 
16:      Store  $d_i$  with  $(s_i, P_j)$  in map
17:    end for
18:  end for
19: end procedure

```

A. Workspace Voxelization

Input to the creation of a robot reachability map procedure is the Unified Robot Description Format (URDF) model of the robot, from which a detailed robot description is obtained. The robot's hypothesized workspace is first voxelized to create a square structure around the robot. The voxelization process is performed by octree [12], a hierarchical data structure enabling spatial partitioning and searching between voxels. The root node of the octree is at the base of the manipulator, and every node is connected to its eight children. The tree is extended to the overestimation of the diameter of the robot arm in an extended state. The size of the voxels required is task dependent and thus user-defined. Tasks requiring a high degree of accuracy will need a smaller voxel size in order to provide

more accurate results in the final base placement location. The centers of voxels are determined, and a sphere with a radius of the voxel's resolution is placed in every voxel.

B. Self Collision Checking

The workspace regions where the robot body is present should not be included in the workspace because end-effectors cannot reach those sections due to collisions. For filtering out such sections, as a preprocessing step, the robot body is modeled as a collection of triangular meshes and checked for collisions with the voxelized sphere centers. The sphere centers in collisions with robot body are opted out of the workspace structure; they are not further discretized for reachability analysis. A collision checking library FCL [13] is used for fast collision checking. This method provides an advantage in terms of time efficiency since most other methods such as [4] and [3], check for collisions only when obtaining solutions from poses. The reachability map structure of three different robots created with the Reuleaux library is shown in Fig. 1, where maps in Fig. 1a and Fig. 1c are created with self-collision checking and Fig. 1b is created without it.

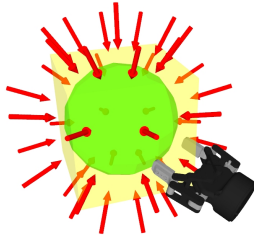


Fig. 2. Discretization of a single voxel in the workspace. The cube (yellow) is the voxel to be searched. A sphere (green) is fitted inside the cube. The sphere is discretized with probable poses (red arrow)

C. Workspace Sampling

Uniform sampling of the joint space does not guarantee uniform sampling in the Cartesian space. For uniformly sampling the task space, the spheres representing the voxels are sampled for uniform point distribution on the sphere by the method presented in [14]. From every point of the distribution, a direction is assumed towards the center of the sphere. As depicted in Fig. 2, an individual voxel of the discretized workspace is fitted with a sphere, and the sphere is sampled for points on its surface. From an individual point, a direction is considered towards the center of the sphere represented by a red arrow. Considering a frame, the direction towards the center of the sphere is the z -axis, while the x and y axes are tangential to the sphere surface.

All the frames created in the previous step are collected and searched for inverse kinematics (IK) solutions. A closed-form analytical inverse kinematics solver IKFast [3] is employed to find inverse kinematics solutions. For manipulators where analytical solutions are not available, a numerical solver KDL [15] is used which is less time-efficient due to the nature of the solver. All reachable poses that return a valid joint solution are stored.

D. Measure of Reachability

The reachability measure of a given voxel can be calculated by:

$$D_{RM} = (R/N) * 100 \quad (1)$$

where number of reachable poses is R and the total number of discretized poses in that sphere is N . Based on the calculated D_{RM} , spheres are assigned to different colors ranging from red to blue in ascending order in color spectrum. The reachability map generation process is simplified in Algorithm 1.

E. Inverse Reachability Map (IRM)

Every pose in reachability map is represented as:

$$T_{global}^{TCP} = \{x, y, z, \rho, \phi, \theta\} \quad (2)$$

where $\{x, y, z\}$ is the position of the TCP (Tool Center Point) and $\{\rho, \phi, \theta\}$ is the orientation represented in Euler angles. Considering every pose in the reachability map to be a transformation matrix (T) from the origin of the robot, inverse transformation T^{-1} on every pose is performed and stored as an inverse reachability map (IRM). All the T^{-1} stored will be later used in the base placement scenario where task poses are defined by the user.

IV. BASE PLACEMENT

In this section, potential base poses for a given task is identified, where the task poses are completely dependent on the user. Task poses can be a discretized trajectory or indication of certain regions the robot has to reach for pick and place task. All the task poses $task_i$ are transformed using transformations T_j^{-1} of the inverse reachability map to create union map of all the potential base poses B_{ij} .

$$B_{ij} = task_i * T_j^{-1} \quad (3)$$

These solutions are only representations of base locations of the manipulators as the reachability maps are created w.r.t to the base of the manipulators. To obtain the potential base locations of the robot, static transformation of the base of the manipulator to the base of the robot needs to be applied to the solutions, which can be identified by the URDF of the robot.

$$B_{ij} = B_{ij} * (T_{robotbase}^{armbase})^{-1} \quad (4)$$

All the potential base poses B_{ij} are considered to be points and stored in an octree structure; they are processed through a nearest neighbour (NN) algorithm that clusters together all poses in the same voxel and been assigned a sphere for each voxel. This procedure is the inverse process of sphere discretization process. In the previous process, poses are obtained from spheres; here the spheres are acquired from transformed poses. All the spheres are associated with a PlaceBase index (D_{PB}) representing the probability of suitable base locations from where the task poses can be reached.

$$D_{PB} = \begin{cases} \frac{N * B_{max}}{B_{max} - B_{min}} * 100, & \text{if } D_{PB} \geq 1 \\ 0, & \text{otherwise} \end{cases} \quad (5)$$

where B_{max} and B_{min} are the maximum and minimum number of possible base poses in a sphere, and N is the number of base poses in the given sphere. The size of the union map is the size of the inverse reachability map times the number of task poses.

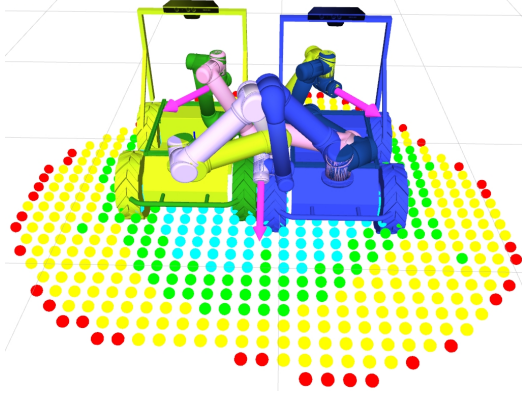


Fig. 3. Two base locations are searched for 3 task poses (magenta arrow). The manipulator configurations from the left robot model (green) are: a) Dark green, b) green, c) white and from the right robot model (blue) are: 1) Dark blue, 2) blue, 3) pink

As the base of most of the wheeled or legged robot models maintain a parallel stance with the ground, the union map can be transferred from 6D Cartesian space $SE(3)$ to 3DOF space $SO(2)$, where the only movement allowed is the rotation about the z -axis and translation in x or y -axes. The union map is sliced through a horizontal axis w.r.t to ground. The 3D union map now can be considered as a 2D union map representing potential base locations on the ground.

In Algorithm 2, m represents the user-defined number of spheres which have the highest PlaceBase index to be searched for base locations. As the orientation is constrained in only one direction, the orientation direction (rotation in z -axis) is sampled though an uniform interval to generate multiple base poses on a single sphere. For all calculated robot base positions, the reachability score of the task poses are considered. From the pool of the base poses which can reach maximum number of task poses, n base poses are considered as final solutions. A typical solution is presented in Fig. 3, where 2 base locations are searched with 3 task poses. Here, the robotic system has a Husky robot as a base, and a UR5 arm as the manipulator. To consider optimal base locations, there should be 6 manipulator states present for 3 task poses and 2 base locations.

V. RESULTS

The Reuleaux map generation library was applied on several robots with different specifications and sizes. Without the self-collision checking, the average reachability map generation

Algorithm 2 Robot Base Location by Inverse Reachability method

input (Inverse Reachability Map, task poses $task_i$, desired number for base locations n , spheres to be searched m) output (n base locations)

```

1: procedure findbaseLocations
2:   for each  $task_i$  do
3:     for each pose  $j$  in IRM do
4:       Find  $B_{ij} * (T_{robotbase}^{armbase})^{-1}$ 
5:     end for
6:   end for
7:   Cluster all  $B_{ij}$  by NN and assign spheres and  $D_{PB}$ 
8:   Slice union map by 2D
9:   for  $m$  spheres with max  $D_{PB}$  do
10:    sample rotations in  $z$  axis uniformly and obtain  $b$ 
11:    Calculate reachability score for each  $b$ 
12:    Find  $n$  base locations with max scores
13:   end for
14: end procedure

```

time is 156s. However, without self-collision, the generated map is just a collection of reachable poses, which is disadvantageous. Since there are no standard metric to measure the efficiency of a reachability map, we can consider time efficiency and generalizability as the metrics. As the reachability map can be generated offline, its utility lies in using the map in other tasks.

TABLE I
REACHABILITY MAP GENERATION

Method	Robot	(x100000) Poses processed	(x1000) spheres created	Time(min)
Reuleaux	PR2	20.938	2552	124.31
	LWR	13.718	5127	160.41
	UR5	7.636	5017	143.27
Diankov et.al [3]	PR2	205.48	-	490.07
	LWR	145.727	-	427.11
	UR5	123.513	-	371.38
Zacharias et al [4]	PR2	104.69	2680	405.47
	LWR	68.59	5213	542.18
	UR5	38.18	4883	413.23

In Table I, an analysis of Reuleaux's reachability map generation method compared to two contemporary methods [3] and [4] on three different robots is presented. PR2 right arm is a 7DOF manipulator, rigidly connected to the robot's body, while the LWR (7DOF) and UR5 (6DOF) are attached to the ground. In method [3], available as open-source library, reachability is not represented as sphere; instead, it is represented as area. The opportunity to set the desired options for creating maps is only limited to saving joint solutions and setting a maximum radius. On the other hand, [4] represents reachability as a sphere and in extension with different shape representations. Because this method is most similar to our approach, we also represent their work as spheres.

For all our experiments, we set the resolution of voxels

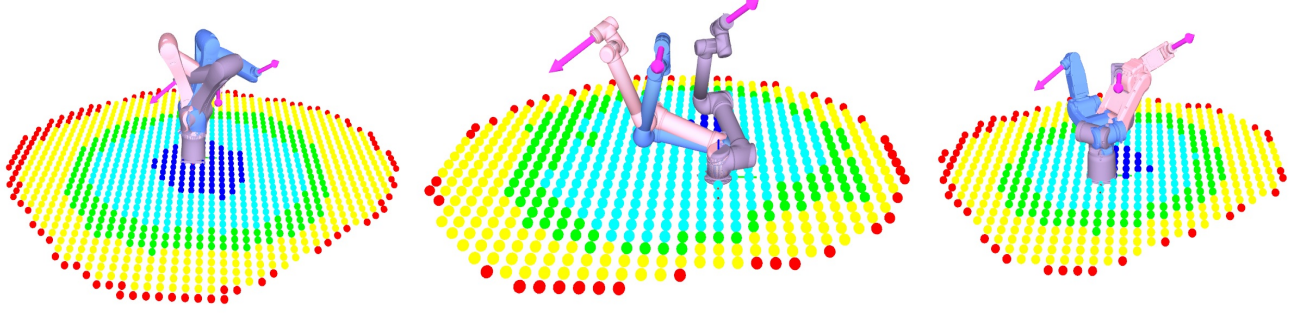


Fig. 4. Base Placement task on 3 different robots. The task is represented by 3 poses in a table environment. The optimal base placement is shown on a) Kuka KR6, b) Universal UR5 and c) Motoman mh5

at $0.08m$ and maximum radius at $1m$. For convenience, our implementation of [4] also uses the same resolution. In Table I, the significant difference in the number of poses processed stems from the fact that [3] obtained the poses by default parameters and in [4] all poses are rotated by 5 degrees in the z -direction to obtain extra poses.

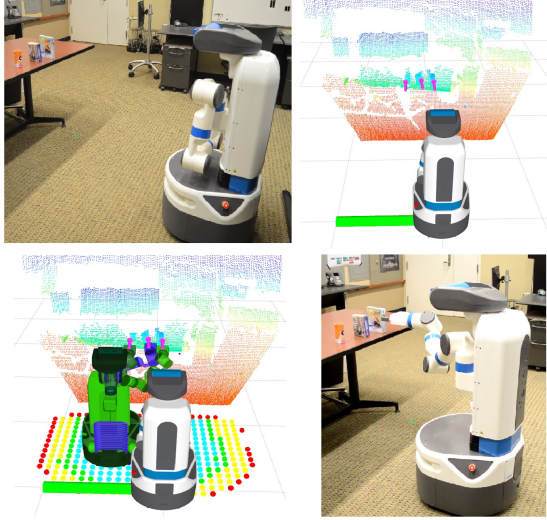


Fig. 5. Task with real Fetch robot. From left to right, top to bottom. a) Real Fetch robot with 3 objects outside of reachable workspace, b) The visualization of the environment with arrows pointing to grasp poses, c) 2D union map with solutions to all task poses, d) The robot is reaching for one of the objects after base placement

For all base placement experiments, task poses are decided based on reach tasks. To represent a motion plan in terms of poses, the trajectory must be uniformly sampled, which is beyond the scope of this paper. In the simulation scenario, an individual task pose represents a region the robot must access. As shown in Fig. 6, the tasks with magenta arrows represent different sections of the kitchen, such as the sink, oven, and drawer. At the initial condition, the tasks were out of robot's reachable workspace. The intent of the system is to find the optimal base location from where all the task poses could be

TABLE II
ROBOT BASE PLACEMENT PERFORMANCE

robot	reachable task	Time(sec)			
		base calculation	soln validation	Reach base	Reach task
PR2 (sim)	4/6	21.8s	0.4s	7.4s	7.1s
	6/6	18.2s	0.7s	5.2s	6.23s
	6/6	19.1s	0.52s	3.1s	2.1s
	5/6	18.6s	0.7s	6.5s	4.78s
Fetch (real)	3/3	9.23s	0.1s	4.12	6.6s
	2/3	8.71s	0.2s	3.2s	7.87s
	1/3	8.23s	0.11s	7.23s	9.23s
	3/3	8.6s	0.13s	4.53s	7.21s

reached. To insert the task poses in the environment, the depth camera situated on the robot is utilized.

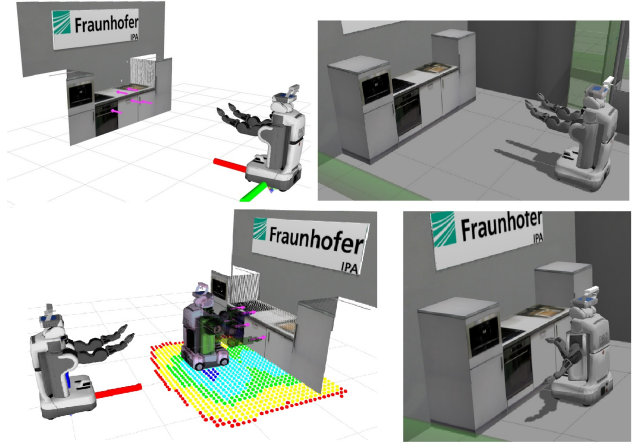


Fig. 6. Base Placement in simulation. From left to right, top to bottom. a) The environment with arrows representing task poses, b) Simulated PR2 robot in a kitchen environment outside of reachable workspace, c) 2D union map with solutions to all task poses, d) Solution execution on the robot after base placement

An evaluation of the robot base placement method, with a comparison with the method presented in [7] and 3 different human users is presented in Table III. The human volunteers are provided with a simple interface where multiple robot

TABLE III
ROBOT BASE PLACEMENT PERFORMANCE

Method	Task Poses	Reachable Solutions	score	Time(sec)
Reuleaux	2	10/10	97.28	1.45
	4	20/20	93.45	2.11
Vahrenkamp et al [7]	2	9/10	89.7	1.77
	4	17/20	81.17	2.64
User1	2	10/10	96.52	1.82
	4	19/20	95.36	2.79
User2	2	9/10	92.8	1.75
	4	19/20	94.22	3.20
User3	2	10/10	97.4	1.84
	4	18/20	79.23	3.11

models can be dragged and set to be final base location solutions. The average scores from 5 tries from the human users are reported. The scores are based on number of task poses reachable from the final solutions.

We also validated our base placement method: (1) in simulations on a PR2 robot in Franhofer IPA Kitchen environment and (2) using a real robot (the Fetch mobile manipulator) in a table environment. For 4 different iterations, the robot is started from different initial locations and the task poses are kept different outside the robot's reach. As result, the robot had to move its base to an optimal base location from where all task poses could be reached. A simplistic base path planner is incorporated to move the robot base.

In both simulation and real environments, the task poses were decided based on the point cloud from the robot's depth camera (Refer to Fig. 5b and Fig. 6b for task poses.) The optimal base location and the 2D union map, along with manipulator joint solutions for individual task poses, are shown in Fig. 5c and Fig. 6c. The final condition, where the robot successfully reaches a task pose is shown in Fig. 5d and Fig. 6d. Table II represents the results from the real world and simulation experiments. In some cases, the robot could not reach the task pose due to failure in motion planning. Since we did not consider environmental obstacle when placing bases, in one scenario, the robot failed to reach 2 of 3 task poses due to collision. Also, since task poses are defined by depth sensors, depth sensor errors had a substantial negative impact on base placement.

VI. CONCLUSIONS

In this paper, methods are proposed to create reachability map of manipulators and find base placement solutions based on the reachability maps where tasks are defined by users. The characteristics that distinguish our process from other available base placement and reachability map creation tool is the time-efficiency and generalizability. Further, the base solution is not limited to a single solution; the numbers of solutions depend on the tasks and user intent. From Table III we can infer that the robot base placement solution presents significantly improved results vis-a-vis human intuition and other methods. It is not possible for a human being to consider the most optimal base locations by intuition. The limitation

of this approach is the input system of the task poses and its exclusion of collision when planning for base placement. The 3D depth cloud sensors are noisy and can provide an incorrect estimation of the environment. Collision in the base placement planning is vital, as the output base pose may be in collision with the manipulated workpiece, worktable, and other surroundings. Also, base placement does not depend only on the reachability of the task poses; the power cost or minimum joint motions should also be considered. The work presented here is available as a self-contained C++ library at <http://wiki.ros.org/reuleaux>.

ACKNOWLEDGMENTS

The authors acknowledge the opportunity provided by ROS-Industrial and Google Summer of Code to accomplish the work presented in this paper. The authors are grateful to Dr. Alba Perez Gracia and Debashree Sheet Makhil for providing encouragement and support throughout the process. We are also thankful to Prof. Maya Cakmak for providing us with a fetch robot we could use for the experiments.

REFERENCES

- [1] J. Miller, U. Frese, and T. Rfer, "Grab a mug - object detection and grasp motion planning with the nao robot," in *2012 12th IEEE-RAS International Conference on Humanoid Robots (Humanoids 2012)*, pp. 349–356, Nov 2012.
- [2] F. Burget and M. Bennewitz, "Stance selection for humanoid grasping tasks by inverse reachability maps," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5669–5674, May 2015.
- [3] R. Diankov, *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.
- [4] F. Zacharias, C. Borst, and G. Hirzinger, "Capturing robot workspace structure: representing robot capabilities," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 3229–3236, Oct 2007.
- [5] T. Yoshikawa, *Foundations of Robotics: Analysis and Control*. Cambridge, MA, USA: MIT Press, 1990.
- [6] N. Vahrenkamp and T. Asfour, "Representing the robot's workspace through constrained manipulability analysis," *Auton. Robots*, vol. 38, pp. 17–30, Jan. 2015.
- [7] N. Vahrenkamp, T. Asfour, and y. p. Rüdiger Dillmann, journal=2013 IEEE International Conference on Robotics and Automation, "Robot placement based on reachability inversion,"
- [8] F. Zacharias, C. Borst, M. Beetz, and G. Hirzinger, "Positioning mobile manipulators to perform constrained linear trajectories," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 2578–2584, Sept 2008.
- [9] F. Zacharias, W. Sepp, C. Borst, and G. Hirzinger, "Using a model of the reachable workspace to position mobile manipulators for 3-d trajectories," in *2009 9th IEEE-RAS International Conference on Humanoid Robots*, pp. 55–61, Dec 2009.
- [10] J. Dong and J. C. Trinkle, "Orientation-based reachability map for robot base placement," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1488–1493, Sept 2015.
- [11] O. Porges, R. Lampariello, J. Artigas, A. Wedler, C. Borst, and M. A. Roa, "Reachability and dexterity: Analysis and applications for space robotics," 2015.
- [12] 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. Software available at <http://octomap.github.com>.
- [13] "FCL flexible collision library," <https://github.com/flexible-collision-library/fcl>. Accessed: 2017-07-11.
- [14] J. Saff and A. Kuijlaars, "Distributing many points on the sphere," in *Mathematical Intelligencer*, vol. 19, pp. 5–11, 1997.
- [15] "KDL kinematics and dynamics library (kdl)," <http://wiki.ros.org/kdl>. Accessed: 2017-07-11.