# Real-time placement of a wheelchair-mounted robotic arm

Pooya Abolghasemi, Rouhollah Rahmatizadeh, Aman Behal, Ladislau Bölöni
Department of Electrical Engineering and Computer Science
University of Central Florida
{pabolghasemi, rrahmati, lboloni}@eecs.ucf.edu, abehal@ucf.edu

***Abstract*— Picking up an object with a wheelchair mounted robotic arm can be decomposed into a wheelchair navigation task designed to position the robotic arm such that the object is "easy to reach", and the actual grasp performed by the robotic arm. A convenient definition of the notion of ease of reach can be given by creating a score (ERS) that relies on the number of distinct ways the object can be picked up from a given location. Unfortunately, the accurate calculation of ERS must rely on repeating the path planning process for every candidate position and grasp type, in the presence of obstacles. In this paper we use the bootstrap aggregation over hand-crafted, domain specific features to learn a model for the estimation of ERS. In a simulation study, we show that the estimated ERS closely matches the actual value and the speed of estimation is fast enough for real-time operation, even in the presence of a large number of obstacles in the scene.**

## I. Introduction

Assistive robotics tries to make the life of disabled or elderly people easier by helping them in the performance of activities of daily living (ADLs). Although most existing technologies such as wheelchair mounted robotic arms are controlled by the user, some level of automation is desired, especially for people with severe physical and cognitive disabilities. Many ADLs involve object manipulation where motion planning is needed for the arm of the robot to reach an object. In these kinds of tasks, positioning the base of the robot is important since it strongly affects the ability of the robot to perform the task. We can divide a manipulation task into two control tasks: the positioning of the robot base TP, and the grasping of the object with the robotic arm TG. These tasks can be executed autonomously or manually based on the preferences of the user. In this paper we try to automate the TP task such that the TG task can be performed effortlessly regardless of autonomous or manual execution.

Let us consider a manipulation task by a mobile robot in which the goal is to grasp an object $c$ in the presence of a set of obstacles $\{o_1, \ldots, o_n\}$. The TG task uses a motion planning algorithm $A_{grasp}$ to find a collision-free path to reach the target. It appears that the task of TP is limited to finding a position from where $A_{grasp}$ can successfully grasp $c$, and thus TP is strongly dependent on the grasping algorithm. For instance if the base of the robot moves to a position from which the arm would not be able to reach the target object, the base should move to another position. Humans, however, have an intuitive understanding that some positions are clearly better than others *without having a specific grasp algorithm in mind*. To transfer this intuition

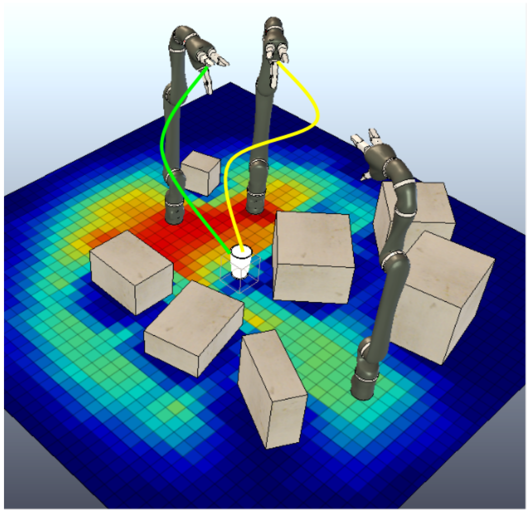


Fig. 1. The arm tries to reach a target object (cup) while avoiding obstacles. The ease of reach score (ERS) is shown by a heatmap in which blue represents low ERS, while red shows a high ERS. The arm can easily reach the target when it is placed on a position with high ERS.

to a robot we define a metric called *ease of reach score* (ERS). In Section III the ERS is defined by considering the number of distinct ways the object can be reached from a given position for the base of the arm. Defining the ERS this way separates the positioning task TP from the grasping task TG and would be useful for both automated and manual execution of the TG task.

Figure 1 shows the ERS for different positions from where an arm might try to reach a target (cup). From two positions a collision-free path to the cup is found and the task would be executed successfully. However, in positions where the ERS is low, the manipulation task might fail with a higher probability. For the remainder of this paper, we assume the arm to be the JACO arm by Kinova Robotics [1], a light-weight robotic manipulator with 6 degrees of freedom. The rehabilitation version of this arm is mounted on a wheelchair to build a navigation-manipulation system used by many users with disabilities.

Calculating the ERS is a computationally expensive process since it relies on solving the motion planning problem for all possible poses and grasp approaches. To overcome this limitation and make ERS suitable for real-time operation, in Section IV we propose a method to quickly estimate the ERS. In Section V we show that this technique provides a good estimate of the ERS value, thus solving the problem of robot base positioning in real-time.

## II. Related Work

Placement of a mobile robot is an important problem since it fundamentally affects the ability of the robot to execute the tasks. One approach to address this problem, introduced by Zacharias, Borst and Hirzinger [2], is to create the *capability map* of the robotic arm. A capability map contains the information describing which regions of the workspace are reachable from what directions. This map would be useful in finding a proper robot placement for execution of workspace linear constrained trajectories [3] [4]. Leidner and Brost [5] used object centric reasoning to find a correct place and time of the movement for the base of the humanoid robot Rollin' Justin for a mobile pick-and-place task.

The inverse reachability approach [6] proposed by Vahrenkamp, Asfour and Dillmann finds suitable base poses to reach a target from a particular orientation. However, when it comes to calculate reachability within a target region instead of a single target pose, this method is difficult to use. A technique to generalize the experience of a successful grasp through exhaustive search from different robot poses is presented in [7] by Stulp, Fedrizzi and Beetz. The result of grasps in different positions are classified and used to extract the best base position for a successful grasp.

In [8] Jamone et al. introduced the *Reachable Space Map* which the robot learns autonomously and online during the execution of goal-directed reaching movements. This map can be used to estimate the reachability of a fixed object and to plan preparatory movements.

Yang et al. [9] proposed a methodology for automatic reaching analysis of wheelchair users in an indoor environment. Their method is based on a simple model of a person sitting in a wheelchair and an efficient motion planner.

The presence of obstacles in an environment opens up new challenges to the existing methods that work based on inverse kinematics without considering obstacles. The capability map changes when an obstacle is close to the robotic arm, because the space needed for the arm such that its end-effector reaches a particular pose might be blocked by an obstacle. More recently, we proposed to estimate the effect of obstacles as a Guassian function [10]. However, we relax this assumption in this work since it limits the accuracy of the method.

## III. Ease of Reach Score

We approach the problem of positioning a mobile robot base (in our case, the wheelchair) by finding the best pose for the base of the arm. Once a proper pose is found for the arm, the robot base can also be positioned accordingly. Considering the complexity of a robot arm with high degrees of freedom and the space it needs to reach a target while avoiding the obstacles, positioning the arm to make the target easily reachable is a significant challenge. In this section, we present a quantitative measure for evaluating how easy it is to reach a target from a particular location of the arm.

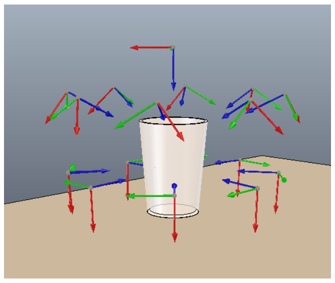


Fig. 2. 17 reachable distinct poses around an arbitrary small target object, here a cup. The Blue arrow shows the direction in which the arm approaches the cup. The poses reachable from the bottom of the cup are blocked by the table.

Let us consider a scenario where a robotic arm positioned at location $(x, y)$ aims to grasp an object $c$ which is located at the origin. Note that the orientation of the base of the arm does not affect the outcome of motion planning since the JACO base joint can freely rotate $360°$. In addition, the height of the arm is fixed as the arm is mounted on a wheelchair. We also assume that the environment contains a number of obstacles $O = \{o_1, \ldots, o_n\}$ that the arm must avoid. The obstacles can be approximated with a cuboid with arbitrary size and pose. In the running example used in this paper, we assume that the obstacles are placed on a table. However, our method can be generalized to situations where the obstacles have arbitrary poses in 3D space. We assume that the knowledge of the algorithm about the environment is perfect, i.e. the shape and pose of the obstacles are known. Note that we did not classify the table in the same group as other obstacles, since the table is an obstacle for the wheelchair and other obstacles are limiting the arm movement and not the wheelchair. The approach we took was to ignore the presence of the table during the training phase. However, the influence of the table was re-added in the estimated ERS by setting the ERS to zero in locations made unreachable by the table.

From a particular position for the arm, the number of collision-free paths to the target $c$ can be zero, for instance when $c$ is out of the reach of the arm. On the other hand, there are positions from where many collision-free paths can be found to reach the target from different directions. To discretize the number of grasp poses, for any small object that is graspable by a robotic hand (such as a cup), we consider two grasps different if the approach angle differs by at least $\pi/4$ as shown in Figure 2. Technically, we defined a region of interest around the target. Defining a region of interest helps us become independent of the target shape, because, regardless of the target shape, as long as the arm can reach many of the poses in the region of interest around the target, it is highly probable that it can successfully grasp the target.

We enumerate the number of ways the arm can grasp the object $c$ from position $p = p(x, y)$ in the presence of the obstacles $O = \{o_1, \ldots, o_n\}$ and call it *Count of Distinct Grasp Trajectories* $CDGT(p, c, O)$. It can be immediately inferred from the definition that obstacles lower or at best keep the CDGT the same, because they make a previously feasible grasp impossible to reach.

$$\forall p \ \forall o_{n+1} \ CDGT(p, c, O \cup o_{n+1}) \leq CDGT(p, c, O) \quad (1)$$

To grasp an object, a robotic arm needs to approach it from one direction, try grasping, and if it fails, have the option to try from another direction. Therefore, it is advantageous to have the ability of approaching the target from different orientations. We want to define the *Ease of Reach Score (ERS)* such that it captures our intuitions about the preferences over different positions $p(x, y)$. We define ERS such that its value would be 0 for positions from where the grasp is not possible, while 1 for the position from which the maximum number of grasp poses are reachable. Starting from these considerations, we will define the ease of reach score as:

$$ERS(p, c, O) = \frac{CDGT(p, c, O)}{\max\limits_{p}\left(CDGT(p, c, \emptyset)\right)} \quad (2)$$

Similar to the CDGT, adding new obstacles to the environment will lower or at best keep the ERS the same:

$$\forall p \ \forall o_{n+1} \ ERS(p, c, O \cup o_{n+1}) \leq ERS(p, c, O) \quad (3)$$

The best position for the arm is the one where the ERS is maximized:

$$p_{opt} = \arg\max_{p} ERS(p, c, O) \quad (4)$$

There are situations where the object is not reachable at all, i.e., the ERS would be zero for all possible positions. In this case, some of the obstacles should be moved in order to reach the target. Another case also might occur in which the optimal position for the arm is not reachable because of the limited number of positions reachable by the wheelchair..

In this paper we do not consider the motion planning problem for finding a collision free path for the base of the mobile robot (wheelchair) from its current position to the desired location. However, one of the advantages of the proposed method is that it not only finds the location of maximum ERS, but it also finds the ERS for all locations. Therefore, other parameters such as the distance to a candidate wheelchair location can also be considered while solving the motion planning problem for the wheelchair. For instance, the final decision might be to accept a position with a smaller ERS value if it offers other advantages (such as easier access by the wheelchair).

## IV. Estimating the ERS

In order to find a collision-free path that puts the end-effector of the arm in a desired pose (one of the defined poses around the target), we use Rapidly-exploring Random Trees (RRTs) [11]. To calculate ERS at each position of the arm we need to solve dozens of motion planning problems (one for each grasp direction). Even by discretizing possible positions for the arm to a very coarse grid (eg. 20 by 20) and considering the desired grasp pose set contains 26 approach poses, $20 \cdot 20 \cdot 26 = 10400$ motion planning problems need to be solved. Note that motion planning is a computationally expensive procedure for an arm with many degrees of freedom. As we show in next section it takes several minutes to calculate ERS using exhaustive search. To overcome this drawback, we propose a method to estimate ERS in order of seconds to make it practical for real-time applications.

Let us assume that the robot arm is located at an arbitrary position $(x, y)$ and its goal is to reach a particular object while avoiding the obstacles. Obstacles block some of the space it needs to reach the target. The blocking effect of an obstacle depends on the size and shape of the obstacle, its distance to the arm, etc. However, we do not know how much an obstacle will affect the ERS at a particular location of the arm. Furthermore, we do not know how the blocking effect of adjacent obstacles close to the target combine together to lower the ERS. We propose an approach that starts from the observation that the ERS for each point in the environment can be estimated considering some general information about the point and its surroundings.

In order to find the value of ERS at a certain location of the arm, we set up a simulation scenario in which a target object is surrounded by 1 to 9 obstacles. The obstacles in the environment are approximated using cuboids with random size and pose. The dimensions of the obstacles can vary between 10-35cm and their distance to the center of target can vary between 10-100cm. By running the simulation multiple times and measuring the ERS at different locations, we gather the information required for the learning process. In order to learn how the ERS is affected in different situations, we hand-crafted a set of features that intuitively capture the aspects of that situation. In other words, these features are designed in such a way to classify the points on a grid with similar ERS values in the same group while distinguishing between the points which have different ERS values.

The features we found useful are illustrated in Figure 3 and explained here:

- *Distance to the closest obstacle (L1).* This feature captures the Euclidean distance between the closest obstacle and the current position of the arm. It is preferred for the arm to be positioned in the furthest possible position from any obstacle. The closest obstacle seems to be the most important one since it probably has the largest effect on ERS.
- *Distance to the target (L2).* The Euclidean distance between the arm and the target which is very important since one of the key factors which affects ERS is how far the arm is located from the target. For instance, when the arm is located far away from the target, reaching the target is not possible, i.e. the ERS is zero.
- *Angle between L1 and L2 ($\theta$).* The angle between L1 and L2 along with the distances L1 and L2 capture the configuration of the arm, target, and the closest obstacle with respect to each other. When the obstacle is located between the target and the arm, the ERS should be lower compared to the situation where the obstacle is located

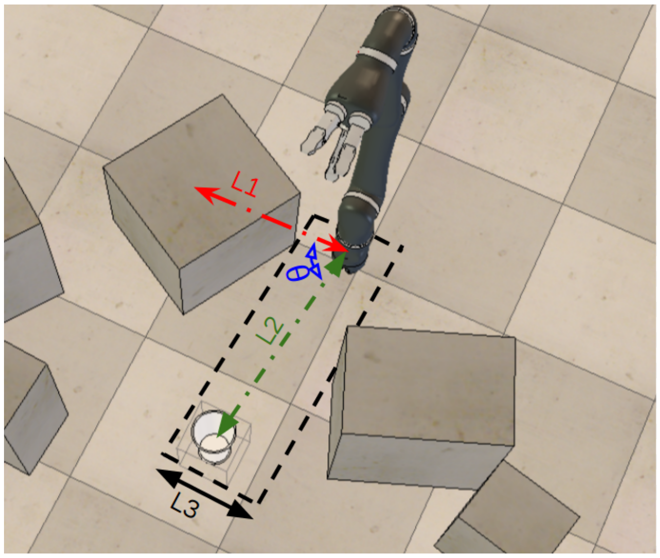


Fig. 3. Demonstration of features in an example scene. L1: distance to the nearest obstacle, L2: distance to the target, L3: widest collision-free path from the arm to the target, and $\theta$: angle between L1 and L2.

outside the space between the arm and the target. This feature tries to distinguish these situations from each other.

- *Widest path (L3).* It measures the width of the widest rectangular obstacle-free path from the arm to the target. This feature tries to estimate how much free space is available for the arm to operate while reaching the target. The wider this path is, the more space the arm has to operate and the result would be a higher ERS.
- *Force field.* In addition to the closest obstacle, other ones also affect the ERS more or less. To estimate the effect of the $n$ closest obstacles, we use a sum of force fields type formula:

$$FF(p, O) = \sum_{i=0}^{n} \frac{Volume(o_i)}{Distance(p, o_i)^2} \tag{5}$$

This formula aims to capture the fact that the larger the size, number and proximity of the obstacles to the target object, the more and more difficult is to find a grasp.

These features had been obtained by experimentally testing an initially larger set and eliminating those that had been found ineffective in reducing the estimation error. In some problems, gathering large number of samples is not easy (e.g. [12]). Fortunately in this domain this is not a problem, thus, we gather enough data by running the simulation 100 times. Then, we calculate each feature for all the points on the grid to be used later in the learning algorithm.

The cardinality of training data is not merely the number of training scenes, but the number of scenes multiplied by the number of candidate arm positions in the scene. For instance, by discretizing the possible positions for the arm into a $(20 \cdot 20)$ grid and repeating the simulation to generate 100 scenes, we gather 40,000 training examples. Calculating the features for each and every point in the environment provides us with a large number of training samples. This is one of the advantages of the proposed approach which makes the learning reasonably fast without requiring too many simulated environments.

The value of the explained features calculated for each training example and the resulted ERS forms the input to the machine learning algorithm. The output would be a predictor able to estimate the ERS based on the features. We use the CART algorithm [13] to create 200 regression decision trees. We use decision trees because of the fact that decision trees properly handle this kind of non-linear problems without relying on a large number of training examples. Finally, to enhance the accuracy of the results, we use the bootstrap aggregating ("bagging") method [14] to predict the value of the ERS.

## V. Results

### A. Estimation accuracy

In this section we validate the proposed approach, by investigating how well the approach solves the positioning problem. We simulate a JACO arm mounted on a wheelchair in V-REP [15] robotic simulator. The techniques proposed in Section IV allow us to calculate the estimated value $ERS_{est}$ while brute force methods allow us to calculate the actual value $ERS_{act}$. Figure 4 shows an example scenario with a target object (cup) and seven obstacles of various sizes and various orientations. The figure on the top shows the actual ERS as a heatmap under the obstacles, while the bottom figure depicts the estimated ERS. A visual inspection of the figures shows that although not perfect, the approximation and the actual value of ERS are very close.

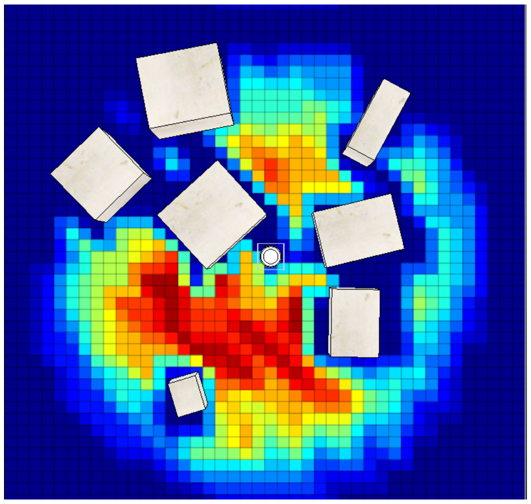
Let us now try to develop a useful error metric. One approach would be to calculate the average error for every grid point. However, this would be a misleading metric, because for large number of locations $ERS$ will be trivially zero (for instance, the ones that are located outside the range of the arm). If we calculate the simple average, the error would depend on how far the grid extends from the origin. Instead, we will define the error metric as being the average only for locations where at least one of the $ERS_{act}$ and $ERS_{est}$ are not zero:

$$P = \{p \mid ERS_{est}(p, c, O) > 0 \vee ERS_{act}(p, c, O) > 0\} \tag{6}$$

We denote the cardinality of this set of points with $\#P$. Thus for a given target object $c$ and set of obstacles $O$ we define the average relevant error $ARE(c, O)$ as follows:

$$ARE(c, O) = \frac{\sum_{p \in P} \left| ERS_{est}(p, c, O) - ERS_{act}(p, c, O) \right|}{\#P} \tag{7}$$

The calculation of the $ARE$ is computationally expensive as it requires the calculation of the $ERS_{act}$. We performed it for 700 test scenes as described in Table I. The first column describes the number of obstacles in the scenes and the second column shows the total number of test scenes with that number of obstacles. The table shows that on average


(a) The actual ERS of the cup.

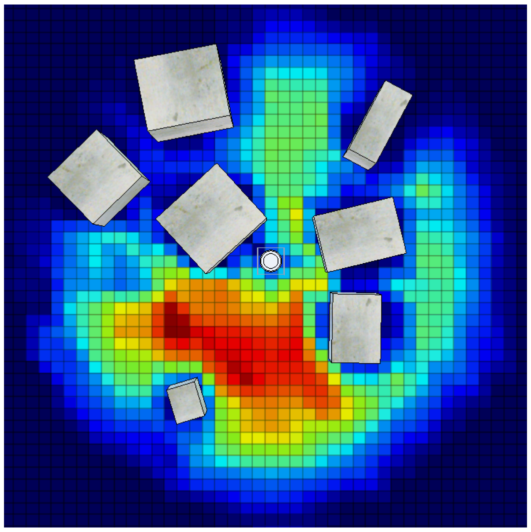

(b) The estimated ERS of the cup.

Fig. 4. An example scene with three obstacles and heatmaps corresponding to $ERS_{act}$ (upper) and $ERS_{est}$ (lower). In the heatmaps, blue corresponds to low and red to high ERS values.

TABLE I
THE AVERAGE RELEVANT ERROR $ARE$ FOR 700 TEST SCENES.

| **Scene description** | **count** | $ARE$ |
|---|---|---|
| 1 obstacles | 90 | 0.0276 |
| 2 obstacles | 79 | 0.0343 |
| 3 obstacles | 81 | 0.0395 |
| 4 obstacles | 73 | 0.0435 |
| 5 obstacles | 67 | 0.0471 |
| 6 obstacles | 73 | 0.0507 |
| 7 obstacles | 73 | 0.0544 |
| 8 obstacles | 73 | 0.0571 |
| 9 obstacles | 91 | 0.0602 |
| | Average | 0.0458 |

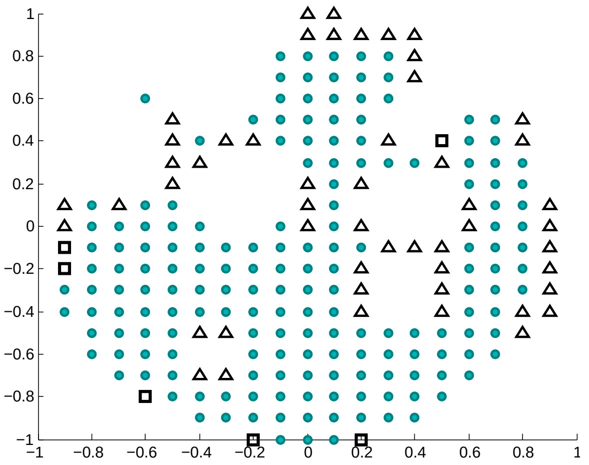


Fig. 5. The distribution of Type 2 and Type 3 errors by type in a scene with three obstacles. The sign ● denotes locations where both $ERS_{act}$ and $ERS_{est}$ are positive. The grid points with no sign are the ones where $ERS_{act} = ERS_{est} = 0$. Type 2 errors are denoted with $\triangle$ while Type 3 errors with $\square$.

the error stays in a moderate range, but in general increases proportional to the number of obstacles.

From the point of view of a practical robot implementation, however, the error in the $ERS$ is not of high importance. The robotic wheelchair needs to position itself such that the object is easily reachable - the finding of the optimal position is of a comparatively small importance. We can divide the errors in the determination of $ERS$ into three major types:

**Type 1:** $ERS_{act} > 0 \land ERS_{est} > 0 \land ERS_{act} \neq ERS_{est}$

If this type of error occurs, the system might choose a position which is not exactly the global optima. Practically, this kind of error is not a major issue as long as the difference between the actual value and the estimated one is not considerable. By trying to keep the total error minimized, the chosen position should fall into a small range of the optimal position.

**Type 2:** $ERS_{act} > 0 \land ERS_{est} = 0$

This issue will happen when the system mistakenly estimates ERS to be zero while the actual ERS is not zero, i.e., reaching the target is possible from that location. As long as the ARE is not a large number, we can assume that these points were likely not a good choice for the arm to do the manipulation. However, in highly congested scenes or in scenarios with constraints on the movement of the wheelchair where there are limited options for the arm to select, the presence of Type 2 errors might make the system mistakenly believe that the problem is unsolvable.

**Type 3:** $ERS_{act} = 0 \land ERS_{est} > 0$

This type of error relates to the positions where the grasp is not possible but the estimation suggests otherwise. As a result, if the wheelchair choose this kind of points to execute the positioning task TP, it would be impossible to perform the grasp task TG. Practically, by performing motion planning for the chosen points, these kind of errors can be avoided. However, this type of error is the most critical one and should be minimized.

Figure 5 depicts the distribution of Type 2 and Type 3 errors in a sample scenario with 7 obstacles. The structure shows that the estimate yielded correct or acceptable values for the majority of positions (but for feasible and unfeasible locations). In a few positions in which Type 2 or Type 3 occurs, all of them located at the boundary between the feasible and unfeasible regions. In practice, the system would choose positions at the interior rather than at the boundary of the feasible zone, thus avoiding both types of errors.

### B. Performance considerations

Let us now consider the performance speedup achieved by our technique. The following table summarizes computational cost of various activities, on a Intel i7-4xxx series system with 16GB of RAM.

| | |
|---|---|
| Generating test data for learning (100 sample scenes) – offline | 14 hours |
| Learning – offline | 2 min 5 sec |
| $ERS_{est}$ – proposed method – online | **0.96 sec** |
| $ERS_{act}$ - naïve method – online | 26 minutes |

We generated 100 sample scenes for the learning process to estimate the ERS. Generating this number of samples normally takes around 43 hours, however, by dedicating each core of the CPU to one simulation process, it took 14 hours. The learning process takes around 3 minutes which is not an issue since it needs to be done only once. We dedicated 700 sample scenes to the evaluation of our method. Calculating the $ERS_{est}$ using the proposed method on each test scene takes only 0.96 seconds on average. On the other hand, the naïve method of finding the actual value of ERS ($ERS_{act}$) at every single location of the arm takes about 26 minutes on average. We conclude that the estimation technique proposed provides a good approximation of the actual ease of reach score ERS for a computational cost four orders of magnitude smaller than the full calculation.

## VI. Conclusions and future work

In this paper, we investigated the problem of positioning a wheelchair mounted robotic arm such that reaching a target object would be easier while avoiding obstacles. First we explained a metric called ease of reach score (ERS) which simply says how easily reachable a target is when the arm is located in a certain position. Then we argued that exhaustive search for finding the ERS at each point is a very computationally expensive process. Therefore, we proposed an alternative, much faster method to estimate the value of ERS using machine learning techniques. By simulating the proposed method in V-REP simulator in which the wheelchair mounted robotic arm tries to reach a target object, we showed that the estimation technique provides sufficient accuracy for practical use.

Future extensions to this work can consider using human demonstrations instead of brute-force trial and error. Reinforcement learning with reward shaping (e.g. [16]) can be used to utilize the sample demonstrations in a more efficient way. In addition, for dealing with the complexity of continuous state space, more advanced techniques (e.g. [17]) can be used instead of naïvely discretizing the state space.

In many cases, the system will find different locations with the same ERS. In this situation, other factors such as the distance between the current location of the wheelchair and each candidate goal should be considered for decision making. Finally, to autonomously navigate the wheelchair to reach the desired pose, recurrent neural networks can be used to generate the movement trajectory [18].

**Acknowledgments:** This work had been supported by the National Science Foundation under Grant Number IIS-1409823.

## References

[1] V. Maheu, J. Frappier, P. Archambault, and F. Routhier, "Evaluation of the JACO robotic arm: Clinico-economic study for powered wheelchair users with upper-extremity disabilities," in *IEEE International Conference on Rehabilitation Robotics (ICORR)*, pp. 1–5, 2011.

[2] F. Zacharias, C. Borst, and G. Hirzinger, "Capturing robot workspace structure: representing robot capabilities," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, pp. 3229–3236, 2007.

[3] F. Zacharias, C. Borst, M. Beetz, and G. Hirzinger, "Positioning mobile manipulators to perform constrained linear trajectories," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, pp. 2578–2584, 2008.

[4] J. Dong and J. Trinkle, "Orientation-based reachability map for robot base placement," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, September 2015.

[5] D. Leidner and C. Borst, "Hybrid reasoning for mobile manipulation based on object knowledge," in *Workshop on AI-based Robotics at IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2013.

[6] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Robot placement based on reachability inversion," in *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1970–1975, 2013.

[7] F. Stulp, A. Fedrizzi, and M. Beetz, "Learning and performing place-based mobile manipulation," in *International Conference on Development and Learning (ICDL)*, pp. 1–7, 2009.

[8] L. Jamone, L. Natale, G. Sandini, and A. Takanishi, "Interactive online learning of the kinematic workspace of a humanoid robot," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, pp. 2606–2612, 2012.

[9] J. Yang, P. Dymond, and M. Jenkin, "Reaching analysis of wheelchair users using motion planning methods," in *Impact Analysis of Solutions for Chronic Disease Prevention and Management*, pp. 234–237, Springer, 2012.

[10] P. Abolghasemi, R. Rahmatizadeh, A. Behal, and L. Bölöni, "A real-time technique for positioning a wheelchair-mounted robotic arm for household manipulation tasks," in *AAAI workshop on artificial intelligence applied to assistive technologies and smart environments (ATSE)*, 2016.

[11] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, pp. 995–1001, 2000.

[12] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, A. Jabalameli, and A. Behal, "Trajectory adaptation of robot arms for head-pose dependent assistive tasks," in *FLAIRS conference*, 2016.

[13] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and regression trees*. CRC press, 1984.

[14] L. Breiman, "Bagging predictors," *Machine learning*, vol. 24, no. 2, pp. 123–140, 1996.

[15] M. Freese, S. Singh, F. Ozaki, and N. Matsuhira, "Virtual robot experimentation platform V-REP: a versatile 3D robot simulator," in *Simulation, Modeling, and Programming for Autonomous Robots*, pp. 51–62, Springer, 2010.

[16] S. A. Raza, B. Johnston, and M.-A. Williams, "Reward from demonstration in interactive reinforcement learning," in *FLAIRS conference*, 2016.

[17] A. Jackson and G. Sukthankar, "Learning continuous state/action models for humanoid robots," in *FLAIRS conference*, 2016.

[18] R. Rahmatizadeh, P. Abolghasemi, and L. Bölöni, "Learning manipulation trajectories using recurrent neural networks," *arXiv preprint arXiv:1603.03833*, 2016.