A real-time technique for positioning a wheelchair-mounted robotic arm for household manipulation tasks

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

Wheelchair mounted robotic arms can help people with disabilities perform their activities of daily living (ADL). The autonomy of such a system can range from full manual control (both wheelchair and robotic arm controlled by the human) to fully autonomous (with both the wheelchair and the robotic arm under autonomous control). Many ADLs require the robot to pick up an object from a cluttered environment - such as a glass of water from a table where several other objects exist. In this paper, we concentrate on the task of finding the optimal position of the base of the robotic arm (which is normally a rigid point on the wheelchair) such that the end effector can easily reach the target (regardless whether this is done through human or robot control). We introduce the ease-of-reach score ERS, a metric quantifying the preferences for the positioning of the base. As the brute force computation of ERS is computationally expensive, we propose an approach of estimating the ERS through a mixture of Gaussians. The parameters of the component Gaussians are learned offline and depend on the nature of the environment such as properties of the the obstacles. Simulation results show that the estimated ERS closely matches the actual value and the speed of estimation is fast enough for real-time operation.

Introduction

Wheelchair-mounted robotic arms, such as the popular Kinova JACO (Maheu et al. 2011) or the Exactdynamics iARM and MANUS arms promise to help disabled or elderly people in the performance of their activities of daily living (ADLs). Such activities involve reaching for everyday objects such as food or drink, personal toiletry, books, eyeglasses and so on. In their early incarnations, such systems were thought of simply as a teleoperated system with the user controlling the wheelchair and/or the arm with a joystick or a similar type of device. However, with the larger penetration of such systems, the robotic wheelchair / robotic arm assembly needs to achieve significant autonomy, bringing such systems into the purview of artificial intelligence.

The desirable degree of autonomy exhibited by such a system is a complex question. The wheelchair-bound users might exhibit various degrees of motor-control or cognitive disabilities. Their disability levels can change with the progression of the disease, rehabilitation or aging. Furthermore, the users might have different preferences for automatic versus controlled behavior, and different levels of trust in the robot (Kim et al. 2012).

Let us consider an ADL scenario where a disabled user, having a motorized wheelchair, with an attached Kinova JACO robotic arm aims to pick up a glass of water from the table. This task can be separated in two main components: the user must move the wheelchair to an appropriate position near the table [1]. Second, the robotic arm must be moved such that it grasps the cup and brings it to the user. Either of these subtasks can be performed under automatic or manual control (and, in fact, the user might make ad hoc decisions about what approach to use). If the user performs both tasks under manual control, in the first phase, the user will aim to get the wheelchair and robotic arm base to a position from where the cup is "easy to reach" - a concept that the users can clearly express preferences about. In the second phase, the user moves the arm to actually grab the cup. This suggests that the "ease-of-reach" metric will be useful for other scenarios as well, for instance when the wheelchair movement is automatic but the grasp manual, or vice versa.

The work described in this paper focuses on the wheelchair positioning part of this task, regardless whether the actual grasp will be done under human or automatic control. In order to do this, we need to (a) quantifying the ease-of-reach in a way that aligns both with human judgement and motion planning algorithms, and (b) find a way to estimate this metric at a speed suitable for real-time operation.

Related Work

The problem of grasping an unknown object in cluttered environments has attracted many AI researchers because of its inherent complexity, for instance, see (Boularias, Bagnell, and Stentz 2015). The problem of the placement of a mobile robot is important since the ability of a robot to execute a task depends on the pose of its base. One approach is

[1] What really matters here is the position of the base of the robotic arm, but this must be achieved through the movement of the whole wheelchair. On the other hand, the JACO base joint freely rotates 360°, thus the orientation is not relevant.

to create the *capability map* of the robotic arm (Zacharias, Borst, and Hirzinger 2007). The information in a capability map describes which regions of the workspace are reachable from what directions. This map can be used to find a convenient robot placement for execution of workspace linear constrained trajectories (Dong and Trinkle 2015).

Machine learning approach to generalize the experience of successful grasp through exhaustive search from different robot poses is presented in (Stulp, Fedrizzi, and Beetz 2009). (Jamone et al. 2012) introduce the concept of a *reachable space map* to address the problem of the robot autonomously learning during the execution of goal-directed reaching movements. (Yang, Dymond, and Jenkin 2012) analyses the reaching power of a wheelchair user based on a simple model of a person sitting in a wheelchair and an efficient motion planner.

The presence of obstacles in an environment creates new challenges to inverse kinematics methods. Similarly, the capability map changes when an obstacle is close to the robotic arm and recreating it is computationally expensive. For instance, in (Stulp, Fedrizzi, and Beetz 2009) the exhaustive search should be repeated since the grasp map of the environment changes by adding an obstacle. In this paper, we present a method based on motion planning which does consider obstacles.

Defining the Ease-of-Reach Score

Let us consider a scenario where a robotic arm positioned at location (x, y) aims to grasp an object c in an environment with obstacles $O = \{o_1, \ldots, o_n\}$ that the arm must avoid. We want to define the *Ease-of-Reach Score (ERS)* such that it captures our intuitions about the preferences over different positions $p(x, y)$. The value of ERS should be 0 for positions from where the grasp is not possible, while 1 for the position from which the grasp can be done under "ideal conditions". As we want to make the ERS independent of the different human or machine motion planning algorithms, we will base our metric on the *number of distinct grasps possible* from a given position. For instance, if from a given position we have 10 different ways to grasp the object, it is likely that this position will be preferred both by the human and the automatically controlled operator. This position would be preferred to one where there is only one possible grasp that the operator would need to get exactly right to successfully complete the task.

Let us now develop a numerical formula for the ERS. We call *Count of Distinct Grasp Trajectories* $CDGT(p, c, O)$ the number of ways the arm can approach an object c to grasp it from base position $p = p(x, y)$ in the presence of the obstacles $O = \{o_1, \ldots, o_n\}$. To discretize the number of grasp poses, we will consider two grasps to be distinct if the approach angle differs by at least $\pi/4$. For the case of our running example, Figure 1 shows 17 distinct grasps for the cylindrical cup. As a note, obstacles lower or at best keep the CDGT the same, because they make a previously feasible grasp impossible to achieve.

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

The ideal condition for a grasp is an environment with no obstacles and a position from where we can choose the largest number of possible grasps. Starting from these considerations, we will define the ERS as:

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

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

$$p_{opt} = \arg\max_{p} ERS(p, c, O) \quad (3)$$

Figure 2 shows the ERS for a scenario with three obstacles and the wheelchair positioned in such a way that the base of the manipulator is at the maximum ERS. Note that the optimal position might not be reachable (due to the fact that the wheelchair on which the robotic arm is mounted has its own limitations, for instance, it might collide with the table).

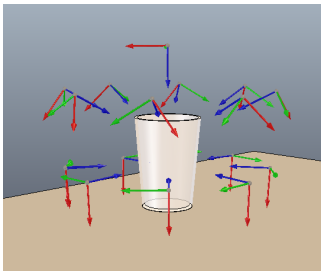


Figure 1: The cup placed on the table can be grasped using 17 reachable grasp poses. The blue arrow shows the direction in which the arm approaches the cup. The grasp poses from the bottom are blocked by the table.

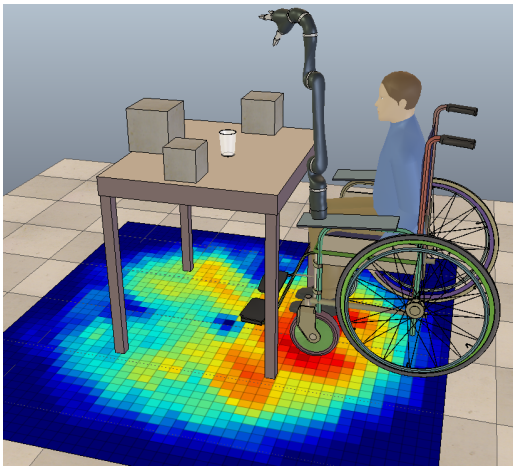


Figure 2: ERS for a scenario with three obstacles. In the heatmap, blue represent low ERS, while red represents a high ERS. The wheelchair is positioned such that the robotic arm is located at the maximum ERS.

Estimating the ERS

The brute-force calculation of the ERS requires us to solve the motion planning problem for every grasp angle and to re-

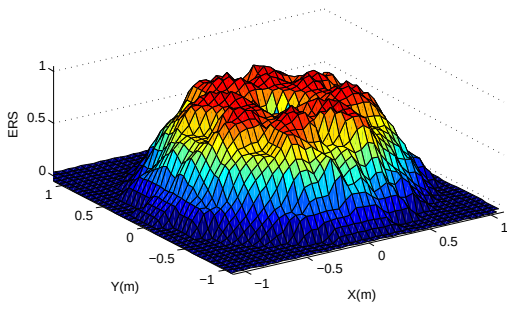


Figure 3: ERS for a small object, here a cup, located in position $p = p(0, 0)$, computed using brute-force computations. Blue: low ERS, red: high ERS.

peat this for every point in a grid covering the possible locations of the robot arm. The computational effort depends on the resolution of the grid, but even a very coarse grid (eg. 20 by 20) yields $20 \cdot 20 \cdot 17 = 6800$ motion planning problems. We use Rapidly-exploring Random Trees (RRTs) (Kuffner and LaValle 2000) to find an obstacle-free trajectory to reach a grasp pose close to the target. Even with this fast method, calculating the exact ERS before every decision is not a feasible approach for a realtime solution of the TP task.

Calculating the ERS offline is feasible if there is no obstacle to consider. For instance, Fig 3 shows the ERS calculated using this method for a cup positioned at $(0, 0)$ without any obstacles around it. As expected, the ERS has a ring shape - the reach is difficult both if the arm originates too far or too close to the object. The maximum ERS, for this setup is reached at the distance of 0.5m from the object. Note that this calculation needs to be done only once and is valid for any small object that is graspable by the robotic arm since ERS is agnostic to the shape of the grasp target. The robot can store this map, and recall it whenever it needs to perform the TP task. However, the ERS also depends on the number, location and size of the obstacles, thus the presence of obstacles leads to a combinatorial explosion of the possible maps. In our setup with n obstacles and 10 different obstacle sizes, the number of maps is $(10 \cdot 20 \cdot 20)^n$, that is $1.6 \cdot 10^7$ for two obstacles and $6.4 \cdot 10^{10}$ for 3 obstacles.

It is thus desirable to find a way to quickly estimate the ERS without the need to compute extensive offline libraries. The approach we propose starts from the observation that the maximum ERS is obtained when no obstacles are present, while each obstacle reduces the ERS.

We shall assume that the ERS-reducing effect of each obstacle can be separated into a *blocking function* $B(p, c, o)$ and these blocking functions take effect independently:

$$ERS(p, c, \{o_1 \dots o_n\}) \approx max\left(0, ERS(p, c, \emptyset) - \sum_{i=1}^{n} B(p, c, o_i)\right) \quad (4)$$

The max construct is necessary to ensure that the ERS conforms to the requirement of returning 0 when the grasp is not possible. Without this construct, in the case of multiple obstacles the value could dip into negative numbers as each obstacle subtracts its blocking function from the optimal ERS value.

Let us now consider the shape of the blocking function $B(p, c, o)$. The first observation is that this function will be *translation invariant* for the simultanous movement of the grasp origin p, the object c and the obstacle o. Second, since the grasp poses defined in ERS and the obstacles are assumed to be symmetric, the blocking function will also be *rotation invariant* for rotations centered on the object c.

With respect to the impact of p we expect the blocking function to be highest for values of $p = o$, and decrease as the position of the base is farther away. Thus, a reasonable approximation can be obtained if we assume that the blocking function is a Gaussian centered on c, expressed as a function of $dist(p, o)$. The magnitude and the standard deviation of the Gaussian, however, will depend on the distance of the obstacle to the target object $dist(c, o)$ and the size of the obstacle $size(o)$:

$$B(p, c, o) \approx f(A, \sigma, p) = A \cdot \exp\left(-\frac{(dist(p, c))^2}{2\sigma^2}\right) \quad (5)$$

where

$$A = T_A(dist(o, c), size(o)) \quad (6)$$

$$\sigma = T_\sigma(dist(o, c), size(o)) \quad (7)$$

With these assumptions, the challenge is to determine the expressions for A and σ. For any particular obstacle we can express the $B(p, c, o)$ value from Equation 4 as follows:

$$B(p, c, o) = ERS(p, c, \emptyset) - ERS(p, c, \{o\}) \quad (8)$$

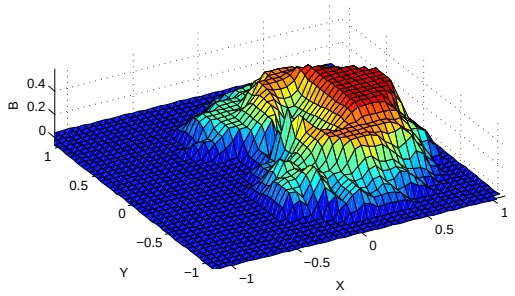
The $B(p, c, o)$ value from Equation 8 can be calculated by brute force strategy (on our grid, it requires $2 \cdot 20 \cdot 20 \cdot 17 = 13600$ motion planning calculations). We can then use a least squares fitting method to find the A and σ values for which the value from Equation 5 most closely approximates the value from Equation 8:

$$A, \sigma = \arg\min_{A,\sigma}\left(\sum_{p}\left(B(p, c, o) - f(A, \sigma, p)\right)^2\right) \quad (9)$$

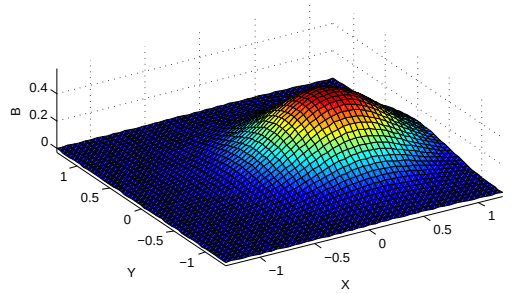
Figure 4 shows an example of this fitting process.

Although with this approach we did not need to exhaustingly consider every combination of multiple obstacles, we still need to calculate for all combinations of obstacle sizes and distances from the target. As we could not calculate all the possible combinations, we run the simulation 1300 times with random obstacle sizes between $10cm$ to $50cm$ and the distance from the target between $0.2m$ and $0.75m$. Finally, we used locally weighted regression (Cleveland 1979) to fit a curve to data containing obstacle features to predict A and σ.

Figure 5 illustrates the resulting values, by showing heatmaps for the evolution of the values of σ (a) and A (b). Fig 5(a) shows that the value of σ decreases as the obstacle is placed further from the target (because by increasing



(a) The value of blocking function $B(p, c, o)$ computed through brute force computation.

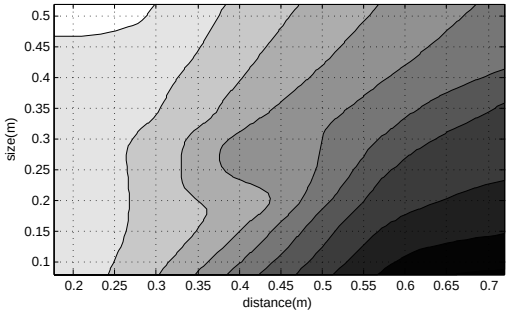


(b) The Gaussian function $f(A, \sigma, p)$ obtained by performing a least-squares fit according to Equation 9.

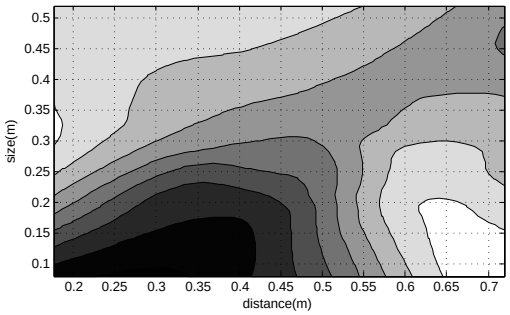
Figure 4: The blocking function of an obstacle and its Gaussian approximation for a small object (cup) located in the origin and an obstacle of the shape of cube of 21cm, located at position $(0.43m, 0.0m)$.

the distance, the obstacle can affect a smaller area around it), and increases with the size of the obstacle. The height A of the Gaussian, shown in Fig 5(b) has a more complex behavior. For large obstacles, the height of the Gaussian decreases with the distance to the target. For small obstacles, however, the height increases with the distance to the target. This behavior is explained if we look at the two graphs together, as shown in the actual shape of the resulting Gaussians in Fig 5(c), which shows that the small obstacles far away from the target will have a blocking function in the shape of a tall but very narrow Gaussian, which only blocks the specific location.

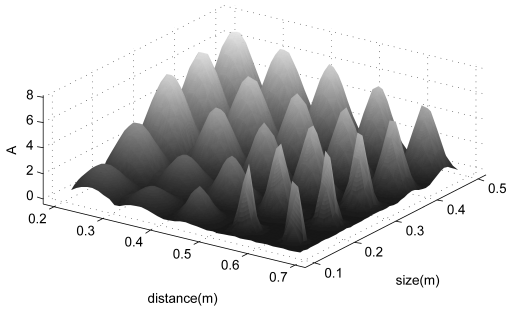
Note that when the obstacle is closer to the target it will affect more points in its surrounding area. In this case, the fitting algorithm tries its best to cover the whole blocking surface to minimize the error, hence, the fitted blocking function's maximum will fall down as it is interpolated with its surrounding points. In Fig 5(c) you can see the maximum A when the obstacle is located further from the target since it affects a smaller area around it. As a result, the maximum for these obstacles are shown as a thin pulse.



(a) The value of σ in the Gaussian approximation of the blocking function. Lighter colors represent a larger value.



(b) The value of A in the Gaussian approximation of the blocking function. Lighter colors represent a larger value.



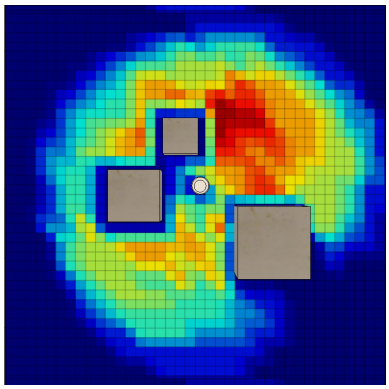
(c) The shape of the Gaussian approximations for various values of the distance and size of obstacles. Note that the width of the individual Gaussians are not on the same scale as the distance and size axis.

Figure 5: Parameters of the Gaussian approximation of the blocking function for A to B.

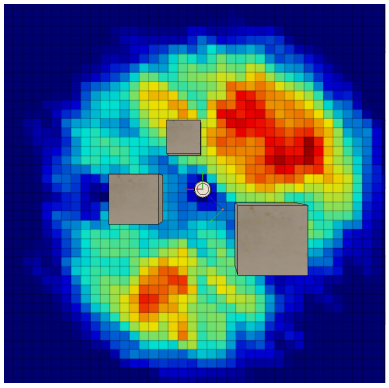
Results

Accuracy

In this section we investigate how well the proposed approach solves the positioning problem by implementing it in the V-REP simulator (Rohmer, Singh, and Freese 2013). First we need to design useful error metrics. Estimation techniques that measure the absolute error in the ERS or B functions are not particularly interesting: we are not interested in specific values of B, only in whether the result al-



(a) The actual ERS of the cup.



(b) The estimated ERS of the cup.

Figure 6: 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.

lows us to solve the positioning problem or not.

Let us first calculate an appropriate error metric for the ERS values. The technique proposed in previous section allows us to calculate the estimated value ERS_{est} while brute force methods allow us to calculate the actual value ERS_{act}. Figure 6 shows an example scenario with a target object (a cup) and three obstacles of various sizes. The upper figure shows the actual ERS as a heatmap under the obstacles, while the lower figure is the heatmap of the estimated ERS. A visual inspection of the figures shows that although not perfect, the approximations are reasonably 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 a large number of locations ERS will be trivially zero (for instance, the ones that are 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} or ERS_{est} is greater than zero:

$$P = \{p \mid ERS_{est}(p, c, O) > 0 \vee ERS_{act}(p, c, O) > 0\} \tag{10}$$

Table 1: The average relevant error ARE for a collection of scenes

Scene description	ARE
Scene 1 (3 obstacles)	0.086
Scene 2 (3 obstacles)	0.091
Scene 3 (2 obstacles)	0.085
Scene 4 (2 obstacles)	0.039
Scene 5 (4 obstacles)	0.128
Scene 6 (4 obstacles)	0.106
Scene 7 (8 obstacles)	0.206

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:

$$ARE(c, O) = \frac{\sum_{p \in P} \left| ERS_{est}(p, c, O) - ERS_{act}(p, c, O) \right|}{\#P} \tag{11}$$

The calculation of the ARE is computationally expensive as it requires the calculation of the ERS_{act}. We performed it for a representative collection of sample scenes as described in Table 1. The table shows that in average the error stays in a moderate range, but in general increases with the number of obstacles.

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. Therefore, we can divide the errors in the determination of ERS into three major types:

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

The practical impact of such an error would be that that system might not choose the optimal position for reaching the target object - under normal circumstances this is a very minor issue.

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

In this case, the system would overlook positions from where the grasp is possible. In most cases, this is not an issue, as these positions were likely not very good anyhow. However, it can be a problem in highly constrained scenarios - for instance when many obstacles limit the number of feasible points and/or constraints on the movement of the wheelchair limit the number of points where the base can be actually positioned. In these cases, the presence of Type 2 errors might make the system mistakenly believe the problem to be unsolvable.

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

These are positions where the estimate believes that a grasp is possible but it turns out not to be the case. If the wheelchair would execute the positioning task TP based on this estimate, it would find that the grasp task TG is impossible from this location.

Figure 7 represents the distribution of Type 2 and Type 3 errors in a sample scenario with three obstacles. The structure shows that the estimate yielded correct or acceptable

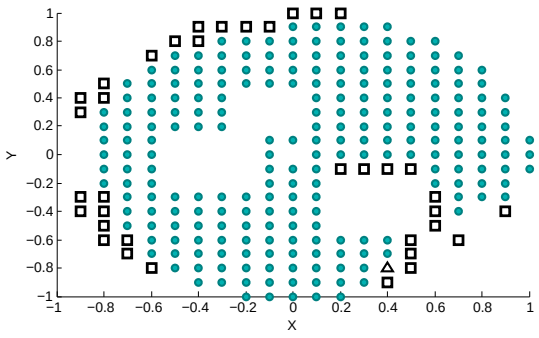


Figure 7: The distribution of Type 2 and Type 3 errors in a scene with three obstacles. The sign • denotes locations where both ERS_{act} and ERS_{est} are positive. The grid points with no sign are the ones where $ERS_{act} = ERS_{est} = 0$. Type 2 errors are denoted with △ while Type 3 errors with □.

values for the majority of positions (but for feasible and unfeasible locations). There is a single Type 2 position and a limited number of Type 3 positions, 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.

Performance considerations

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

Generating test data for learning (1300 sample scenes) – offline	5 days
Learning – offline	2 min 5 sec
ERS_{est} – proposed method – online	**0.82 sec**
ERS_{act} - exhaustive search – online	26 minutes

We find that the learning of ERS estimation is computationally expensive – not so much for the learning process itself, which takes around 2 minutes, as the generation of the single obstacle ERS values that form the basis of learning. This process took about 5 days. Once the values are learned, the calculation of the ERS_{est} takes only 0.82 seconds, in contrast to the computation of the ERS_{act} that takes about 26 minutes in average.

Conclusions

In this paper, we considered the task of positioning a wheelchair mounted robotic arm in preparation for the grasping of a target object in the presence of obstacles. We introduced the ease-of-reach score ERS as a metric for the suitability of certain positions for the base of the robotic arm. As the calculation of the ERS map for an environment is computationally expensive, we proposed an approximation technique based on modeling the ERS as a combination of optimal ERS and the blocking functions corresponding to the obstacles. Through the implementation of the proposed system for the case of a cylindrical cup target object and symmetrical obstacles, we have shown that the resulting approximation is sufficiently accurate for practical use and represents a four magnitude decrease in computational cost compared to the computation of the actual ERS.

References

Boularias, A.; Bagnell, J. A.; and Stentz, A. 2015. Learning to manipulate unknown objects in clutter by reinforcement. In *Proceedings of AAAI Conference on Artificial Intelligence*.

Cleveland, W. S. 1979. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association* 74(368):829–836.

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

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

Kim, D.-J.; Hazlett-Knudsen, R.; Culver-Godfrey, H.; Rucks, G.; Cunningham, T.; Portee, D.; Bricout, J.; Wang, Z.; and Behal, A. 2012. How autonomy impacts performance and satisfaction: Results from a study with spinal cord injured subjects using an assistive robot. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on* 42(1):2–14.

Kuffner, J. J., and LaValle, S. M. 2000. RRT-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, 995–1001. IEEE.

Maheu, V.; Frappier, J.; Archambault, P.; and Routhier, F. 2011. 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)*, 1–5.

Rohmer, E.; Singh, S. P.; and Freese, M. 2013. V-REP: A versatile and scalable robot simulation framework. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and System (IROS)*, 1321–1326. IEEE.

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

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

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