Minimizing the number of iterations when computing a base pose for manipulation by mobile base inclusion in the inverse kinematics

Janno Lunenburg, René van de Molengraft and Maarten Steinbuch
Department of Mechanical Engineering, Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
E-mail: {j.j.m.lunenburg,m.j.g.v.d.molengraft,m.steinbuch}@tue.nl

Abstract— **One of the most fundamental skills of a domestic service robot is the ability to manipulate objects. Hereto, the end-effector must be positioned accurately in Cartesian space. To move the end-effector to its desired pose, a suitable end-pose for the mobile platform and a corresponding configuration of the manipulator and possibly torso must be found. To determine the optimal base pose, it is common to compute manipulator configurations corresponding to a large number of possible base poses, requiring a large number of IK computations with each a number of samples or iterations with corresponding time-consuming collision checks. This paper demonstrates how the total number of required iterations can be minimized by computing a single IK solution for the kinematic chain including the base kinematics and using the result as the base pose for manipulation. As a secondary objective, the distance to joint limits is maximized. A further reduction in the total number of iterations can be achieved by only constraining translations and rotations that need to be constrained for the task at hand.**

I. Introduction

Over the past years, increasing research attention has been devoted to domestic service robots. These robots usually consist of a mobile base with one or two robotic arms. These arms have six or seven Degrees-of-Freedom (DoFs) and are possibly mounted on a torso with additional DoFs. One of the most fundamental skills of a domestic service robot is the ability to manipulate objects. Accurately positioning the end-effector in Cartesian space is crucial to this end. To move the end-effector to its desired pose, a suitable base pose and a corresponding configuration of the manipulator and possibly torso must be found. Here, the base pose is defined as the position x, y and orientation θ of the robot on the ground plane with respect to a fixed reference frame. Commonly, the base pose and manipulator configuration are computed separately because:

- The kinematic and dynamic characteristics of platform and manipulator usually differ significantly [1];
- A relatively large part of the end-effector positioning error is due to the mobile platform [2];
- Subsequent motion planning algorithms are also performed decoupled, reducing a high dimensional motion planning problem into two problems of lower dimension.

The research leading to these results has received funding from the Dutch ministry of Economic affairs.

To determine the optimal base pose, it is common to compute the manipulator configuration for a large number of base poses. The pose which gets the lowest score on a predefined cost-function is subsequently selected. This cost-function concerns secondary objectives such as distance to joint limits, manipulability and distance to obstacles. Examples of this approach are [2], [3], [4], [5]. References [2] and [3] search a grid, where every grid cell represents a possible base pose. An Inverse Kinematics (IK) solver computes a corresponding manipulator configuration for the cell where the optimization is started and its eight neighbors. The cell for which the resulting manipulator configuration has the highest manipulability is then selected until a (local) minimum has been reached. By starting this optimization from 10 to 20 randomly selected initial poses, the global optimum is found. In [4], a co-evolutionary algorithm is used with IK computed for the manipulator only. In [5], a generalized success model is learned offline, which is subsequently used online to map positions to a predicted probability distribution for successful manipulation. However, the learned success model might not always be applicable to the situation at hand.

These approaches all require a large number of IK solutions to be computed since a manipulator configuration is determined for every evaluated base pose. Depending on the used algorithm (see Section II), each IK solution requires a number of samples (in case of a algorithmic search-based approach) or iterations (in case of a numerical iterative solver). If a collision-free solution is required, this implies that a large number of collision-checks needs to be performed. This is a drawback of these approaches since collision-checking is computationally the most expensive step in the optimization process [4], [6]. Therefore, it is desired to minimize the total number of samples or iterations required to optimize the base pose for manipulation.

This paper demonstrates how the total number of required iterations can be reduced by computing a single IK solution for the kinematic chain including the DoFs of the base. As a secondary objective, the distance to the joint limits is maximized. A number of situations is used to illustrate the effectiveness of this approach.

In the next section, the base pose optimization is discussed. The results will be presented in Section III, followed by a discussion and concluding remarks.

II. Optimizing the base pose for manipulation

The main purpose of this work is to compute the base pose that is optimal for manipulation using as few iterations or samples as possible. As mentioned in the introduction, this is done by including the DoFs of the base in the kinematic chain of the manipulator. It is assumed that torso joints, if present, are also included. Computing IK solutions plays a central role in this approach for determining the optimal base pose. These solutions can be computed in various ways, as will be discussed next, followed by the a description of the algorithm that is used in this approach, the way redundancy is exploited and how the base can be included efficiently.

A. Inverse Kinematics Algorithms

Since the kinematic chains of domestic service robots such as the PR2 [7], the Care-O-bot 3 [8], [9], and AMIGO have more than six DoFs, there is no analytical solution due to the redundant DoFs. As an alternative, numerical solvers are used. A *numerical iterative IK solver* based on the generalized pseudo-inverse of the Jacobian matrix corresponds to the Newton method for a system of nonlinear equations [10]. Such solvers have a few distinct advantages:

- they offer flexibility in selecting the DoFs that need to be constrained, *e.g.*, when grasping an object which is axisymmetric around the $z-$axis this DoF does not need to be constrained.
- redundancy can be addressed straightforwardly by projecting the contributions of secondary objectives, such as distance to joint limits, optimization of manipulability or distance to obstacles, into the Jacobian nullspace.

An alternative way of computing IK solutions is introduced in [11], which uses an *algorithmic search-based approach*. Although this has proven to be a robust and fast approach it also has some drawbacks:

- It has a fixed set of inverse kinematics parameterizations (Transformation, Translation, Rotation and Look-at Ray). This means it is not possible to specify a goal pose in different DoFs;
- Redundancy is addressed by setting a selected free joint to a user specified value when solving the IK. This means that the dimension of the search space increases with every DoF. For example, for a 7-DoF manipulator the dimension of the search space is $7-6=1$, for a 8-DoF manipulator this is $8-6=2$ etc. Furthermore, the entire search space should always be evaluated to find the optimal solution, requiring many samples in case of multiple redundant DoFs.

Due to the flexibility of selecting the DoFs that need to be constrained and the exploitation of redundancy, the numerical iterative solver will be used, which is discussed in more detail in the following sections.

B. Differential kinematics of redundant manipulators

The numerical iterative IK solver makes extensive use of the Jacobian matrix $\mathbf{J}(\mathbf{q})$, which maps the joint space velocities $\dot{\mathbf{q}}$ to Cartesian space velocity $\mathbf{v}_e$:

$$\mathbf{v}_e = \mathbf{J}(\mathbf{q})\dot{\mathbf{q}} \tag{1}$$

with $\mathbf{v}_e$ the $r \times 1$ vector of end-effector velocity of concern for the specific task and $\mathbf{J}$ the corresponding $r \times n$ Jacobian matrix. Furthermore, $\dot{\mathbf{q}}$ denotes the $n \times 1$ vector of joint velocities. With a redundant manipulator, there are $n-r$ redundant DoFs. With $n \times n$ positive definite weighting matrix $\mathbf{W}$ it can be shown that:

$$\dot{\mathbf{q}} = \mathbf{W}^{-1}\mathbf{J}^T\left(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^T\right)^{-1}\mathbf{v}_e \tag{2}$$

Redundancy can be utilized to optimize secondary objectives incorporated in cost function $H(\mathbf{q})$. Eq. 2 is then extended to:

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\dot{\mathbf{v}}_e - \mathbf{N}\mathbf{W}^{-1}\left(\frac{\partial H}{\partial \mathbf{q}}\right)^T \tag{3}$$

with $\mathbf{J}^{\dagger}$ the weighted generalized pseudo-inverse $\mathbf{J}^{\dagger} = \mathbf{W}^{-1}\mathbf{J}^T\left(\mathbf{J}\mathbf{W}^{-1}\mathbf{J}^T\right)^{-1}$ and $\mathbf{N}$ the null space projection matrix $\mathbf{N} = \left(\mathbf{I} - \mathbf{J}^{\dagger}\mathbf{J}\right)$. The weighting matrix $\mathbf{W}$ is used to tune the contributions of the various degrees of freedom. Although a *weighted* generalized pseudo-inverse is well-known in linear algebra, this is often not applied in IK solvers.

C. An iterative numerical inverse kinematics algorithm

This relation can now be used in a closed-loop solution scheme. With $\mathbf{e} = \mathbf{x}_d - \mathbf{x}_e$ the error of the end-effector in Cartesian space, *i.e.*, the difference between the desired end-effector pose x_d and the current end-effector pose x_e, and $\mathbf{K}$ a positive definite matrix, (3) changes to:

$$\dot{\mathbf{q}} = \mathbf{J}^{\dagger}\mathbf{K}\mathbf{e} - \mathbf{N}\mathbf{W}^{-1}\left(\frac{\partial H}{\partial \mathbf{q}}\right)^T \tag{4}$$

which can be numerically integrated with step size Δt:

$$\mathbf{q}(t_{k+1}) = \mathbf{q}(t_k) + \dot{\mathbf{q}}\Delta t \tag{5}$$

This update step is repeated until the termination condition is met: the norm of the error $\mathbf{e}$ has decreased below a certain threshold, in this case 1×10^{-5}.

To enforce a solution between the joint limits of the manipulator, an additional check is performed on every joint:

$$q_i(t_{k+1}) = \min(q_{i,\max}, \max(q_{i,\min}, q_i(t_{k+1}))) \tag{6}$$

Evidently, truncating the update step of one or more joints deteriorates performance and therefore it is convenient to keep the joints as closely as possible to the center of their range, as will be shown in Section III-D.

D. Exploiting Redundancy

As discussed above, redundancy in manipulators can be used to optimize various secondary objectives. In this case, the distance to the joint limits is maximized. Hereto, the cost function:

$$H = \sum_{i=1}^{n} k_i \frac{\left(q_i - q_{i,0}\right)^2}{\left(q_{i,\max} - q_{i,\min}\right)^2} \tag{7}$$

is introduced [12], with k_i a gain, $q_{i,\min}$ and $q_{i,\max}$ the joint limits and $q_{i,0} = (q_{i,\max} + q_{i,\min})/2$. The partial derivative with respect to joint q_i is given by:

$$\frac{\partial H}{\partial q_i} = \frac{2k_i}{(q_{i,\max} - q_{i,\min})^2} (q_i - q_{i,0}) \tag{8}$$

As becomes clear from this expression, this particular choice for joint limit avoidance costs acts as a spring on each joint to keep it in the center of its range.

Note that the cost function H is not included in the convergence criterium, hence the algorithm might terminate while H has not yet reached its optimum. Nevertheless, adding this to the convergence criterium would increase the required number of iterations which is undesirable. Although exploiting redundancy this way has already been introduced by [13], this concept is not yet by default used in IK solvers. Addressing redundancy this way is more common in case of Cartesian control of robots where no explicit IK solution is computed such as, *e.g.*, [14], [15].

E. Including the base platform in the kinematic chain

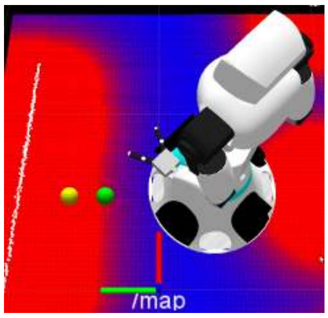

Fig. 1. The 2D costmap is shown where blue is free space and red defines (inflated) obstacles, resulting from the sensor measurements indicated by the white line. The base pose is defined with respect to the "/map" frame as indicated in the figure. The spheres indicate how an update step that would result in a collision (yellow sphere) is truncated (green) to remain collision-free.

To compute the optimal base pose, the two translational and one rotational DoFs of the base are added to the kinematic chain. These are defined with respect to a fixed frame, see Fig. 1. An important difference between the base on the one hand and the manipulator and torso on the other hand is the notion of joint limits. In case of the base platform, these are not defined but the base pose might be constrained by the environment. This can be seen in Fig. 1: the obstacle on the left side of the figure, indicated by the white line representing sensor measurements, is inflated with the radius of the smallest circle that circumscribes the robot footprint and a safety margin. The safety margin forms a constraint while selecting the optimal base pose for grasping: increasing the safety margin decreases the solution space. The inflated obstacle is shown in red in the costmap. An update step in the optimization process might result in the center of the base to invade the red area, which can be detected by querying the costmap. This is computationally of low cost compared to a collision check. The yellow marker indicates the situation where an update step invades the red area. If this is the case, a bi-section algorithm is used to limit the step in this direction, *i.e.*, the step in x- and y-direction is truncated to the point that the base is just *not* in collision, as is indicated by the green marker.

As mentioned in Section II-B, the matrix $\mathbf{W}$ can be used to weigh the contribution of the various controlled DoFs. Choosing the weighting factors for the base large with respect to the manipulator joints will cause the base to converge quickly; nevertheless, if an obstacle is encountered and the contribution of the base is truncated, the convergence will deteriorate significantly because the manipulator joints do not contribute significantly. This illustrates the importance of weighting the generalized pseudo-inverse in an iterative algorithm.

III. Results

As mentioned previously, the aim of the approach described above was to use as few iterations as possible to compute the optimal base position for manipulation. To illustrate the use of the base positioning algorithm, two cases are discussed in Sections III-A and III-C where the AMIGO robot has to grasp an object which is initially out of reach. Furthermore, the effect of the joint limit avoidance will be discussed in Section III-D.

AMIGO is the domestic service robot of the Eindhoven University of Technology. Its base platform has four omni-wheels and is hence fully holonomic. It is equipped with two 7-DoF Philips Experimental Robotic Arms. These have the dimensions of the arms of a large person and are placed on an extendable upper body. In its lower position, AMIGO can grasp objects from the floor while in its upper position it has the size of a child and can therefore operate most features in a domestic environment. The kinematic structure of AMIGO is redundant: including the base platform, the kinematic chain contains $3 + 1 + 7 = 11$ DoFs.

A. Grasping an object from a table

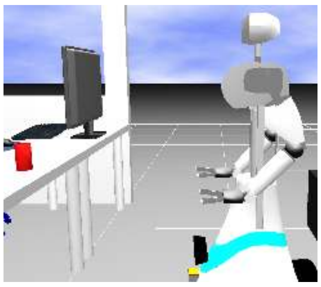

Fig. 2. The AMIGO robot must driver closer to the table to be able to grasp the red can.

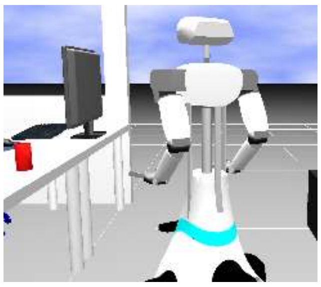

Fig. 3. In this position, the can is within reach.

The first situation that is discussed is shown in Fig. 2. Here, the AMIGO robot has to grasp the red can, which is currently out of reach, from the desk. The optimization was executed with various initial base poses. Note that the robot will use its left gripper, with the grasp vector perpendicular to the edge of the table.

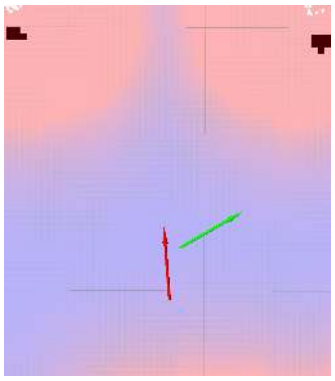

Fig. 4. Initial condition (red arrow) and optimized pose (green arrow) corresponding to the situations in Figures 2 and 3. Similar to Fig. 1, the blue area is free and red defines inflated obstacles. The black areas represent the table legs in the localization map, while the white dots indicate the measured position of the table legs.

The results of this optimization are shown in Figures 3, 4 and 5. It can be seen that convergence of the optimization is always reached within 80 iterations. To compare this result of *one* optimization with 80 iterations: [2] typically performs more than 100 optimizations (since an IK solution is computed for each grid cell), each requiring multiple iterations. Similarly, [4] used a population size of 84 and hence required a minimum of 84 optimizations. Therefore, the total number of iterations for both algorithms is at least an order of magnitude larger than the approach presented in this paper. As can be seen in the lower plot of Fig. 5, the joint limit costs have all converged to approximately the same value.

B. Experimental Results

The situation described in Section III-A has also been investigated experimentally. The results of this experiment are comparable to the simulations, as can be seen by comparing Figures 4 and 6.

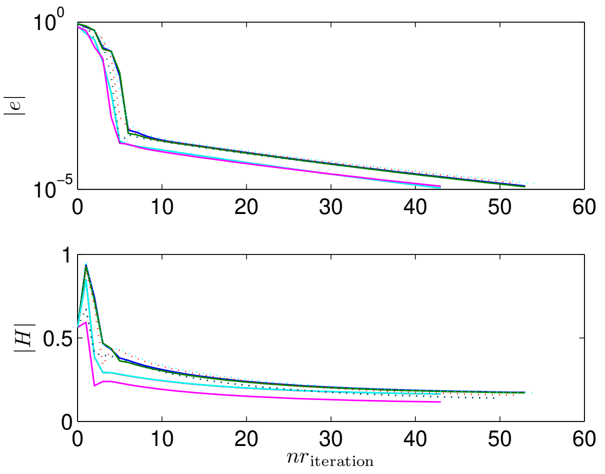

Fig. 5. Norm of the error (upper plot) and joint limit avoidance cost (lower plot) while optimizing the base pose with different initial conditions.

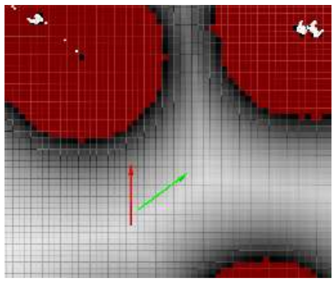

Fig. 6. Initial condition (red arrow) and optimized pose (green arrow) corresponding to the situations in Figures 7 and 8. Here, the red area defines inflated obstacles, resulting from sensor measurements (the white dots) of the table legs.

In Figures 7 and 8, the robot can be seen in its initial pose and in its final pose. Here, it can also be seen that the final base pose is similar to the simulation result (see Fig. 3) and that the robot is indeed able to grasp the object from the table at this base pose.

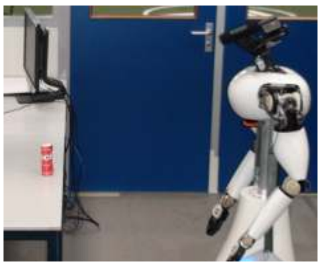

Fig. 7. The AMIGO robot must driver closer to the table to be able to grasp the red can.

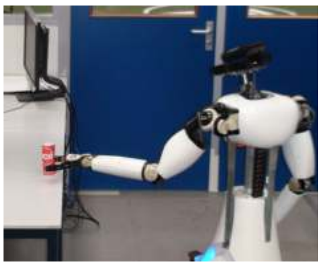

Fig. 8. In this position, the can is within reach.

C. Releasing degrees of freedom

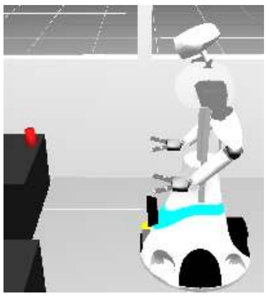

Fig. 9. AMIGO must grasp the can with its left hand: what should be the orientation of the gripper and the corresponding base pose?

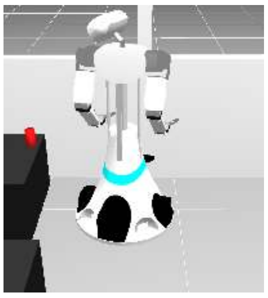

Fig. 10. In this position, the can is within reach.

The situation in Sections III-A and III-B is relatively easy: the can is placed sufficiently far from obstacles such as the table legs. The situation in Fig. 9 is much more challenging since the object is placed close to a wall and the robot cannot move its base underneath the table. This situation becomes even more challenging if the robot has to grasp the object with its left gripper. Commonly, a number of grasp vectors is defined which are all evaluated, *i.e.*, base poses and corresponding manipulator configurations need to be computed for every vector, leading to a dramatic increase of the required collision checks. In this case, four grasp vectors have been defined for which the base pose is computed (see Fig. 11). The convergence and joint limit costs are shown with dashed lines in Fig. 12. Furthermore, the corresponding joint values of the manipulator are displayed in Fig. 13. It appears that the solutions differ significantly, both in convergence as well as joint limit costs H. The magenta solution does converge quickly, but the cost H is significantly larger that for the other angles. This can be attributed to joint wrist_2, which is close to its joint limit. Based on the cost criterium H, one would choose the green solution.

Nevertheless, the can is axisymmetric, hence instead of evaluating a discrete number of possible grasp vector, one DoF (in this case the yaw angle) can be left unconstrained in the optimization. The vectors $\mathbf{v}_e$ and $\mathbf{e}$ now have length 5 instead of 6 while the Jacobian matrix $\mathbf{J}$ correspondingly only has 5 rows. Instead of computing the base pose four times and selecting the best one, only one optimization with 89 iterations is required (indicated by the green grasp vector in Fig. 11 and the solid line in Fig. 12). This shows the reduction in the total number of iterations that can be obtained when properly selecting the DoFs that need to be constrained. The joint limit costs have not converged entirely but the difference from the sampling-based solution is negligible.

D. Joint limit avoidance

The optimizations discussed in the previous sections have all been performed using the joint limit avoidance as discussed in Section II-D. To illustrate the effect of joint limit avoidance, the optimization of Section III-C is repeated with increasing contribution of the joint limit avoidance algorithm. The results are displayed in Fig. 14, where the blue lines indicate the situation without joint limit avoidance and the green, red, cyan, magenta and brown lines indicate increasing contribution.

It appears that having no joint limit avoidance leads to very poor performance, also in terms of convergence. This can be explained by looking at the joint values (see Fig. 15).

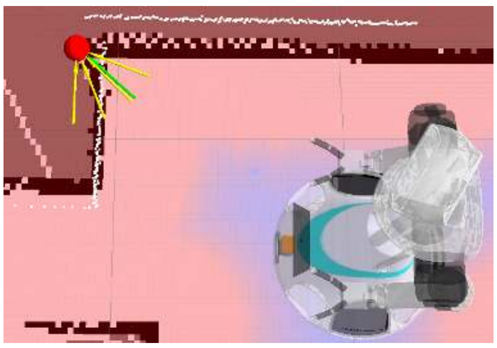
Fig. 11. The yellow arrows indicate the constrained grasp vectors for the red object, corresponding to Fig. 9. The green arrow indicates the optimized grasp vector.

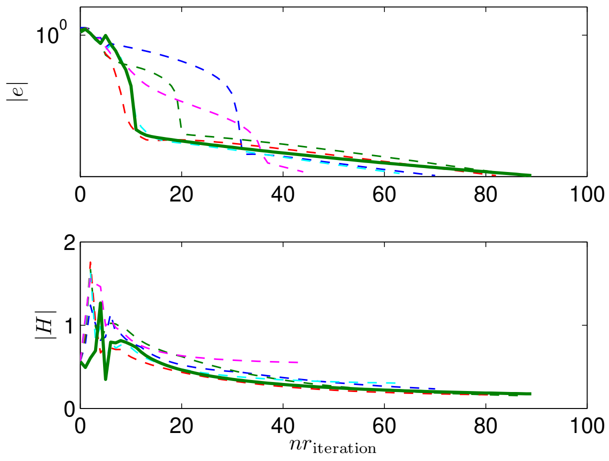

Fig. 12. Norm of the error (upper plot) and joint limit avoidance cost (lower plot) for different grasp vectors (the yellow vectors in Fig. 11). The solid green line indicates the situation where the end-effector yaw is left unconstrained (the resulting grasp vector is the green vector in Fig. 11).

It appears that after some excursions during the first ten iterations, the torso joint, $\mathrm{shoulder}_3$ joint and elbow_1 joint have moved to their joint limits. As a result, the update steps of these joints are truncated, deteriorating the rate of convergence. If joint limit avoidance is applied, the solutions moves away from the joint limits and the convergence rate is restored. This can be seen by looking at the green and red lines: after a number of samples, the error $|e|$ quickly converges below the threshold 1×10^{-5}.

Nevertheless, the joint limit avoidance parameters should not be chosen too large: if the update step $\Delta\mathbf{q}$ is too large, the Jacobian $\mathbf{J}(\mathbf{q})$ changes significantly hence the joint limit avoidance does not map in the Jacobian nullspace anymore. The magenta and brown solutions in Fig. 14 already require more iterations, indicating that there is a tradeoff between convergence and optimality in terms of joint limit avoidance.

As a conclusion, it can be stated that actively avoiding the joint limits prevents the optimization from moving into the limits of the solution space as much as possible, improving convergence. Furthermore, the maximized distance to joint limits will be an advantage if a pose error is introduced when actually positioning the base.

IV. Discussion

In this paper only collisions of the base platform are considered; hence, a 2D costmap can be used to this end. Nevertheless, the ultimate goal is to find a collision-free base pose for which a corresponding collision-free manipulator configuration exists. Similar to [14], one can define pairs of closest points p_i between robot segments and between robot segments and external objects with associated distance d_{p_i}, the cost g_{p_i} per collision pair p_i is given by:

$$g_{p_i} = \begin{cases} s\left(d_{p_i} - d_B\right)^2 & 0 \leq d_{p_i} \leq d_B \\ 0 & d_{p_i} > d_B \end{cases} \qquad (9)$$

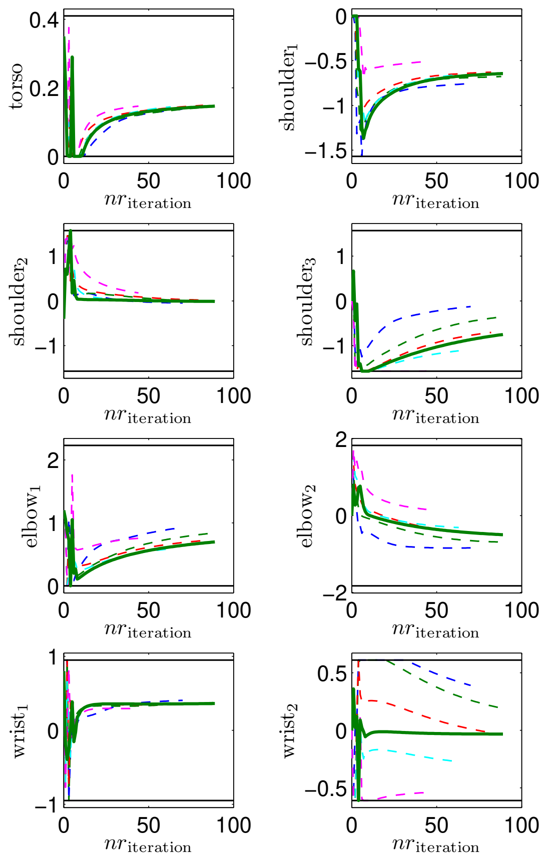

Fig. 13. Joint positions with the end-effector yaw constrained (dashed) and unconstrained (green, solid). The black lines indicate the various joint limits.

where d_B denotes a threshold below which the costs are zero and s denotes the slope. The total costs are given by:

$$H_{\text{collision}} = \sum_{i=1}^{P} g_{p_i} \tag{10}$$

Implementation of this approach is future work.

One of the main concerns using an iterative numerical algorithm as discussed in Section II is the question whether the solution converges within a certain number of iterations. Convergence may be slow or end up in a local minimum, something that might also happen in the approach presented here. One of the ways to prevent this is selecting proper initial conditions. In [2], [3], 10 to 20 initial poses are used to compute one base pose. In the situations discussed in Section III, a random initial pose has been used, with the constraint that the robot should have the object to grasp somewhere in front of him. The number of iterations can be further reduced by selecting a proper initial base pose. A pragmatic approach hereto is based on workspace analysis of the manipulator. This has indicated that grasping at offsets $x_{\text{offset}} = 0.5$ m and $x_{\text{offset}} = 0.2$ m has the highest

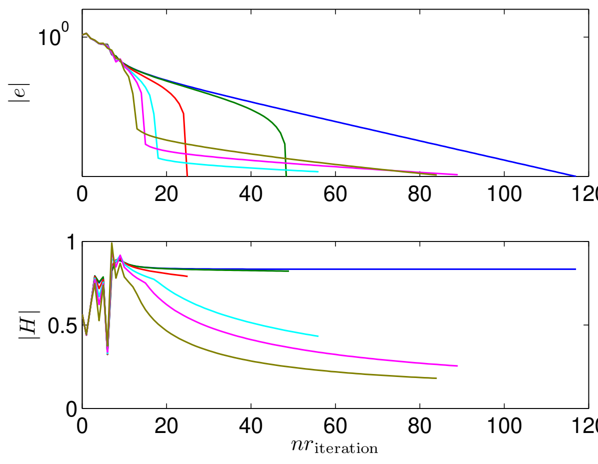

Fig. 14. Norm of the error (upper plot) and joint limit avoidance cost (lower plot) for increasing contribution of joint limit avoidance.

probability of succeeding. To compute the initial pose, the costmap is sampled a number of times on a circle with radius $r = \sqrt{x^2_{\text{offset}} + y^2_{\text{offset}}}$. The sample with the lowest cost is subsequently selected as the initial pose. The orientation also follows directly from both offsets. The result is shown in Fig. 16, where red arrows indicate that the base would be in collision with the environment, yellow and green arrows indicate feasible positions of which the green one has the largest distance to obstacles. By starting the optimization with a distance and orientation that is suitable for manipulation, as well as sufficient clearance from obstacles, convergence is less likely to be slow or end up in a local minimum. The green arrow therefore represents a suitable initial base pose to speed up the optimization process.

V. Conclusions

The purpose of this paper was to compute a suitable base pose for manipulation using as few iterations as possible. This was achieved by including the DoFs of the base platform in the kinematic chain. This way, a suitable base pose could be determined using a single IK computation, using less than 100 iterations.

This is a great improvement compared to other methods; these compute manipulator configurations for at least 80 possible base poses, each requiring multiple iterations. The total number of iterations for these methods is hence at least an order of magnitude larger than for the approach presented in this paper. Since collision checking is the most time-consuming part of grasp-planning, minimizing the number of iterations and hence the number of collision checks will ultimately speed up grasp planning.

Furthermore, it was shown that maximizing the distance to joint limits can improve convergence, which will also be an advantage if a pose error is introduced when actually positioning the base.

In future work, additional secondary objectives such obstacle avoidance can be added to this approach similar to the

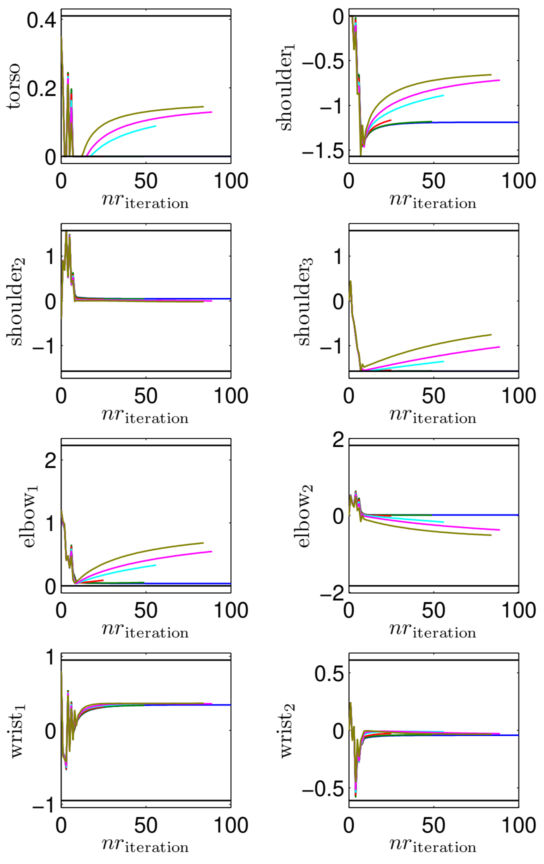

Fig. 15. Joint positions corresponding to Fig. 14.

currently implement joint limit avoidance. A further decrease in the required number of iterations can be achieved by suitably selecting the initial pose.

References

[1] K. Nagatani, T. Hirayama, A. Gofuku, and Y. Tanaka, “Motion planning for mobile manipulator with keeping manipulability,” in *2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 2, Lausanne, 2002, pp. 1663–1668.

[2] K. Yamazaki, M. Tomono, and T. Tsubouchi, “Pose planning for a mobile manipulator based on joint motions for posture adjustment to end-effector error,” *Advanced Robotics*, vol. 22, no. 4, pp. 411–431, 2008.

[3] K. Yamazaki, M. Tomono, T. Tsubouchi, and S. Yuta, “Motion planning for a mobile manipulator based on joint motions for error recovery,” in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Beijing, 2006, pp. 7–12.

[4] D. Berenson, J. Kuffner, and H. Choset, “An optimization approach to planning for mobile manipulation,” in *2008 IEEE International Conference on Robotics and Automation*, 2008, pp. 1187–1192.

[5] F. Stulp, A. Fedrizzi, and M. Beetz, “Action-related place-based mobile manipulation,” in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, St. Louis, MO, 2009, pp. 3115–3120.

[6] D. Ojdanic and A. Graser, “A fast motion planning for a 7dof rehabilitation robot,” in *2007 IEEE 10th International Conference on Rehabilitation Robotics*, 2007, pp. 171–178.

[7] K. Wyrobek, E. Berger, H. Van Der Loos, and J. Salisbury, “Towards a personal robotics development platform: Rationale and design of an intrinsically safe personal robot,” in *Proceedings of the 2008 IEEE International Conference on Robotics and Automation*, Pasadena, CA, 2008, pp. 2165–2170.

[8] B. Graf, M. Hans, and R. Schraft, “Care-O-bot II - development of a next generation robotic home assistant,” *Autonomous Robots*, vol. 16, no. 2, pp. 193–205, 2004.

[9] B. Graf, C. Parlitz, and M. Hägele, “Robotic home assistant Care-O-bot®3 product vision and innovation platform,” in *Proceedings of the 13th International Conference on Human-Computer Interaction, HCI International 2009*, 2009, pp. 312–320.

[10] B. Siciliano, L. Sciavicco, L. Villani, and G. Oriolo, *Robotics, Modeling, Planning and Control*. Springer-Verlag, 2009.

[11] R. Diankov, “Automated construction of robotic manipulation programs,” Ph.D. dissertation, Carnegie Mellon University, August 2010.

[12] Y. Nakamura, *Advanced Robotics: Redundancy and Optimization*. Addison-Wesley Publishing Company, Inc., 1991.

[13] A. Liégeois, “Automatic supervisory control of the configuration and behavior of multibody mechanisms,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 7, no. 12, pp. 868–871, 1977.

[14] M. Behnisch, R. Haschke, and M. Gienger, “Task space motion planning using reactive control,” in *23rd IEEE/RSJ 2010 International Conference on Intelligent Robots and Systems*, Taipei, 2010, pp. 5934–5940.

[15] A. Dietrich, T. Wimböck, A. Albu-Schäffer, and G. Hirzinger, “Reactive whole-body control: Dynamic mobile manipulation using a large number of actuated degrees of freedom,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 2, pp. 20–33, 2012.

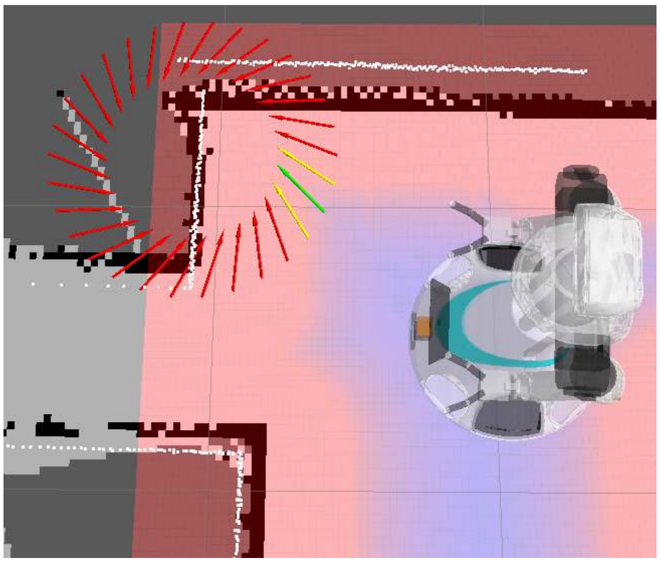

Fig. 16. A pragmatic approach to compute initial conditions: the costmap is queried on a circle at a pre-specified radius from the object of interest. The pose with the lowest cost, indicated with the green arrow, will be used as initial condition for the optimization.