IK–MAP: An Enhanced Workspace Representation to Support Inverse Kinematics Solvers

Nikolaus Vahrenkamp, Dominik Muth, Peter Kaiser, and Tamim Asfour

***Abstract*— We present an approach to improve the performance of general purpose inverse kinematics (IK) solvers which are based on iterative gradient descent algorithms. The proposed *IK-Map* is used to represent the whole workspace of the manipulator through a voxelized data structure, similar to existing approaches, e.g. reachability or capability maps. We extend the reachability map approach by additionally storing reference IK solutions, which can be used to seed iterative IK solvers during online processing. This information can be used to limit the effect of well-known issues with local optimization schemes based on gradient decent methods, such as local minima or constraint violation. We evaluate the approach with a simulated model of ARMAR-4, showing that classical generic Jacobian-based IK solvers can be improved in terms of success rate, performance, and quality of the resulting IK solutions.**

I. INTRODUCTION

Solving the inverse kinematics (IK) problem is an essential capability that is needed for almost every application in the context of robotic manipulation and locomotion. A large variety of components, such as modules related to grasping, motion generation or footstep planning, rely on the possibility to compute joint angle configurations to achieve a specific goal, e.g. a Cartesian pose to be reached by an end effector. In many cases additional constraints, such as self-collision avoidance or pose quality requests, must be considered, which affect the computation complexity. Finally, multiple objectives may be combined in order to generate whole-body postures, e.g. for a humanoid robot.

Several approaches can be used for solving the inverse kinematics problem. Analytic algorithms are fast but they have to be developed specifically for the manipulator. Further it may be difficult to adapt them according to different sets of constraints. Hence numerical approaches are often used to create whole body postures while considering arbitrary constraints. Such methods, e.g. Jacobian Transpose [1], Damped Least Square [2] or hierarchical formalisms [3], [4], [5], usually try to iteratively move the end effector towards the goal until a sufficient accuracy is reached. Due to the iterative nature of the approaches, issues arising from singularities and local minima have to be addressed.

Iterative methods strongly rely on the initial configuration (seed) that is used to start the IK search. The success rate usually decreases with increasing distance to the starting pose, even when IK solutions exist. Stochastic approaches, e.g. Monte Carlo methods, can be used to generate random seeds when the initial process gets trapped in a local minima. Although the success rate increases with such approaches, the results are not deterministic. Further, unnatural postures may be generated which is undesirable e.g. in human-robot interaction tasks.

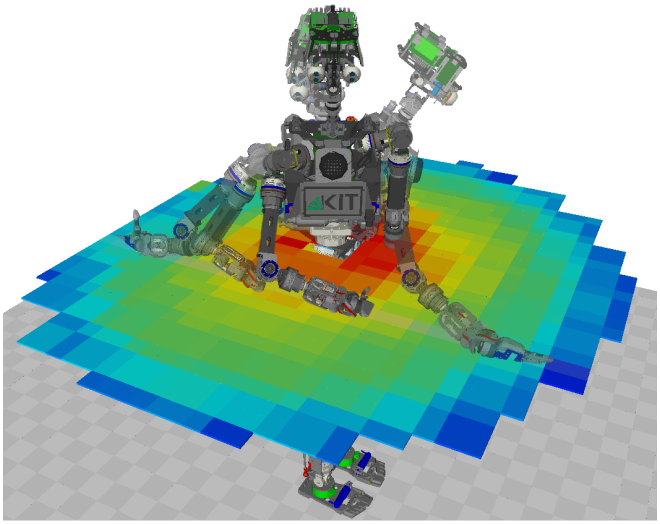

Fig. 1. The *IK-Map* contains reachability data together with reference IK solutions that can be used to support iterative IK solvers. Several reference IK solutions are depicted for a 10 DoF kinematic chain the of the humanoid robot ARMAR-4 covering 2 hip and 8 arm joints.

In this work we address the problem of seeding IK solvers in order to be able to generate whole-body postures for the whole workspace of the robot while considering multiple constraints in an efficient way. Therefore, we present the *IK–Map* approach which provides a representation of the robot's 6D workspace, filled with reference configurations and quality information. In contrast to existing workspace reachability and capability maps [6], [7], [8], the *IK–Map* can be used to directly support IK solvers by providing potential starting configurations (seeds) that can be explored by the IK search.

Although such reference configurations don't give a guarantee that a solution can be found by an iterative IK solver, in particular when varying constraints such as collisions with a changing environment have to be considered, efficiency and success rate can be considerably improved, as we evaluated in this work.

The authors are with the Institute for Anthropomatics and Robotics, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany, vahrenkamp,asfour@kit.edu

The research leading to these results has received funding from the European Union Seventh Framework Programme under grant agreement no 611909 (KoroiBot) and grant agreement no 611832 (WALK-MAN).

II. RELATED WORK

Inverse Kinematics for redundant manipulators is a well investigated topic. In literature, several surveys on IK computation give a good overview on available approaches (see e.g. [1] or [9]). Beside analytical methods (e.g. [10], [11]), which are capable of computing closed-form solutions, iterative methods are widely used since they can be realized in a generic and robot-independent way. Most approaches rely on approximating an inverse of the robot's Jacobi matrix. Several methods, such as the Jacobian Transpose, Damped Least Squares (DLS), or Selectively Damped Least Squares (SDLS) and several extensions are known [12], [2], [13]. More advanced IK approaches are capable to consider multiple equality or inequality constraints which can be ordered in a hierarchy [4], [5], [14]. In [15] pre-designed reference postures are used to support the online generation of whole-body IK solutions. The style base inverse kinematics system [16] is based on a learned probability model of human poses and allows real-time IK solving while reproducing the posture characteristics of an observed human. The IK-fast algorithm [7] can be used for kinematic chains with up to 6 DoF. Kinematic structures with more degrees of freedom can be handled by selecting a subset of active joints, while fixating the remaining ones. Although the efficiency is impressive, the usage for larger kinematic chains is limited.

In [17] a method for generating workspace densities for discretely actuated robots is developed. Due to the discretized configuration space, a tree-based structure can be built and refined during online processing. In order to cope with high dimensional configuration spaces, appropriate discretization parameters have to be chosen, which may result in approximation artefacts. The resulting workspace densities are used to efficiently solve IK-related queries. The concept of building workspace densities is refined in [18], in order to generate a probabilistic measure of the accuracy of the reachable workspace. Similar to the approach of [19] the *workspace quality* is related to the number of configurations that map to a specific area in workspace, which results in misleading representations of singularities (a large number of configurations map to the same area which results in a high quality value, whereas singularities should be avoided). Further, the discretization of the configuration space introduces approximation errors, which could decrease the quality of the results.

Workspace discretization techniques can be used to build a representation of the robot's reachable workspace [6], [8], [20]. These representations are generated offline and usually quality information is stored that encode the capability of the robot's end effector to reach a Cartesian target. The quality information may cover reachability [6], manipulability [19] or stability [21] that may be achieved at the corresponding pose. Usually each discretized 3D or 6D workspace grid cell holds a single value that describes the maximal or average quality value that is achievable inside the extends of the cell. Although this information already gives a good hint of where to search for solutions, it is not guaranteed that a corresponding IK solution can be found, in particular when iterative IK solvers are used. Since such IK algorithms strongly rely on the initial seed, an IK query may fail or result in a low-quality result even when good solutions exist. Therefore we present the *IK–Map* approach, which is used to additionally store reference IK solutions in each workspace cell in order to serve suitable seeds for iterative IK methods (see Fig.1). As we will show, this increases the success rate of iterative constrained IK search as well as it reduces the time needed to search IK solutions.

III. IK–MAP

Initially, the reachable 6D workspace of the robot's manipulator is discretized and filled by an offline procedure. During online processing, this data structure can be queried in order to select reachable targets or to get supporting information that is used to improve the IK solvers for a specific target.

A. Buildup

The *IK-Maps* are generated for a specific kinematic chain and an end effector of the robot. The resulting discretized representation of the reachable 6D workspace is stored as a tree structure, where each node represents a bounding volume in 6D pose space. Similar to an octree in 3 dimensional space, each 6D cell can be seen as hyperoctant that is decomposed into 2^6 children, whereas each of the 6 dimensions is bisected. This bisection is performed until the extends of the grid cell falls below a discretization threshold. Compared to array representations which are usually used for reachability approaches, this tree structure has the advantage that the number of leaf cells is limited to the reachable area. This is critical, since leaf cells hold much more data in our case: instead of storing just one quality value (e.g. 1 byte), a cell additionally contains a reference configuration which is stored with at least $4 * n_{dof}$ bytes (assuming that a floating point value is represented with 4 bytes).

Further, the data structures are created only on write access (i.e. only when a pose is added to the tree), which means that unreachable poses are not represented in order to allow a memory efficient storage. Similar to related works [19], [20], we use a randomized approach for building up the data structures. This has the advantage that a sufficient representation can be built quickly, while iterative refining, even during online processing, is still possible. The generation advances towards a complete coverage as time increases. However, the approach initially leads to an incomplete representation due to the random component. This can be overcome by post processing steps, e.g. Gaussian smoothing or by validating the non-reachability of cells with IK-related methods, similar to [8].

1) Parameters: A pose in the workspace is represented by a 6-dimensional vector, which consists of three entries describing the position followed by three entries that describe the orientation of the pose. Hence, the elements of the workspace tree consist of 6-dimensional vectors which describe the center of the grid cells which may be further

refined by the children. In the following we derive the height of the tree, whereas all leaf cells have the same extend.

The size of the reachable area $l_w \in \mathbb{R}^3$ can be derived from the maximum extends of the workspace $w^-, w^+ \in R^3$:

$$l_w = w_e^+ - w_e^-. \tag{1}$$

Together with the position and orientation discretization parameters $d_p, d_o \in \mathbb{R}^3$, the height of the tree k can be calculated as follows:

First, the number of levels for the position part of the tree $p \in \mathbb{R}^3$ are calculated with

$$\begin{aligned} \frac{l_w}{2^p} &< d_p \Leftrightarrow \\ \frac{l_w}{d_p} &< 2^p \Leftrightarrow \\ log_2(\frac{l_w}{d_p}) &< p \Leftrightarrow \\ p &= \left\lceil log_2(\frac{l_w}{d_p}) \right\rceil. \end{aligned} \tag{2}$$

Second, the number of levels for the orientational part of the tree $o \in \mathbb{R}^3$ are calculated with

$$\begin{aligned} \frac{2\pi}{2^o} &< d_o \Leftrightarrow \\ \frac{2\pi}{d_o} &< 2^o \Leftrightarrow \\ log_2(\frac{2\pi}{d_o}) &< o \Leftrightarrow \\ o &= \left\lceil log_2(\frac{2\pi}{d_o}) \right\rceil. \end{aligned} \tag{3}$$

Finally, the height $k \in \mathbb{N}$ is determined as

$$k = \max\{p_1, p_2, p_3, o_1, o_2, o_3\}. \tag{4}$$

2) Generation: The map is generated as shown in Algorithm 1. The *IK-Map* $\mathcal{I}$ is initialized with the kinematic description of the manipulator $\mathcal{M}$. Then, $\mathcal{I}$ is continuously updated with random samples. For each randomly sampled configuration $\mathbf{c} \in \mathbb{R}^{n_{dof}}$, the quality $q \in \mathbb{R}$ and the workspace pose $\mathbf{p} \in \mathbb{R}^6$ of the end effector is computed. If the grid cell that corresponds to $\mathbf{p}$ has not been set or, if the corresponding quality value is lower than the current q, the entry is updated, whereas $\mathbf{c}$ and q are stored within the data structure. The quality q is evaluated by using the extended manipulability measure as proposed in [22], but other methods for determining a quality value for a given configuration can be incorporated as well. It is also possible to store multiple quality measures to perform a task dependent selection during online processing. An exemplary map for a 10 DoF kinematic chain of ARMAR-4 [23] can be seen in Fig. 2. The figure depicts the best achievable orientation at each 3D grid position whereas the orientation of the end effector is visualized by arrows. The color encodes the stored quality value (blue: low, red:high).

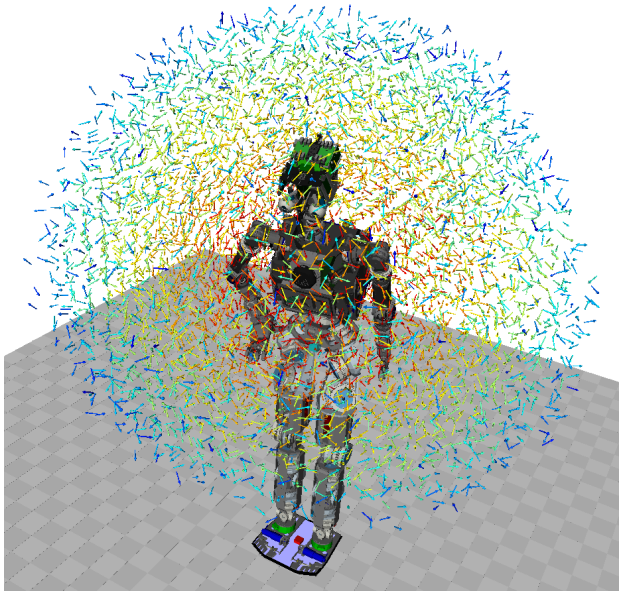

Fig. 2. An *IK–Map* for the 10 DoF kinematic chain of ARMAR-4 covering 2 hip and 8 arm joints. The manipulability is encoded by color ranging from blue (low values) to red (high values).

Algorithm 1 BuildIkMap

Require: QualityMeasure $\mathcal{Q}$, Manipulator $\mathcal{M}$

$\mathcal{I} \leftarrow InitializeIkMap(\mathcal{M})$
while $(!Timeout())$ **do**
  $\mathbf{c} \leftarrow RandomConfiguration()$
  $q \leftarrow \mathcal{Q}.computeQuality(\mathbf{c})$
  $\mathbf{p} \leftarrow \mathcal{M}.forwardKinematics(\mathbf{c})$
  if $(!\mathcal{I}.hasEntry(\mathbf{p}) \,||\, \mathcal{I}.getQuality(\mathbf{p}) < q)$ **then**
    $\mathcal{I}.addPose(\mathbf{p}, \mathbf{c}, q)$
return $\mathcal{I}$

B. Query

During online processing, several components, such as task planners, motion generation modules or footstep planners, may request the quality that can be achieved at a given pose. In addition, IK solvers might be queried in order to generate potential configurations which place the end effector at a requested pose in workspace. A query to the *IK–Map* data structure can be processed with $O(k)$, whereas k is the height of the tree. The tree height depends on the extends of the workspace and the discretization parameters (see Section III-A.1).

The *IK–Map* is used in Algorithm 2 to support the IK query by using the stored reference configuration that is associated with the workspace pose $\mathbf{p}$. Therefore the *IK–Map* tree structure is queried in order to retrieve the grid cell $\mathbf{v}$ in which $\mathbf{p}$ is located. The corresponding configuration $\mathbf{q}_{seed}$ is used as a starting configuration for the iterative IK solver $\mathcal{IK}$. Since the reference value $\mathbf{q}_{seed}$ results in an end effector pose $\mathcal{M}.forwardKinematics(\mathbf{q}_{seed})$ that lies within the extends of $\mathbf{v}$, the upper bound for the position displacement that the IK solver needs to address is d_p, while

the orientational displacement is at maximum d_o. The result is checked against self collisions to ensure collision-free IK solutions. This step is necessary since although it is ensured that the reference configuration $\mathbf{q}_{seed}$ does not result in self-collisions, the resulting configuration $\mathbf{q}$ might be in collision with the robot or the environment.

Algorithm 2 ComputeIK($\mathbf{p}$)

Require: IKMap $\mathcal{I}$, IK-Solver $\mathcal{IK}$

```
v ← I.getGridCell(p)
q_seed ← v.getReferenceConfig()
IK.setSeed(q_seed)
q ← IK.solve(p)
if (!CollisionFree(q)) then
    return NULL
return q
```

Since the IK query may fail, even when seeding with a promising reference, the IK search can be extended as shown in Algorithm 3. When the direct IK call fails, the reference configurations of all $3^6 - 1$ neighboring cells of $\mathbf{p}$ are used to seed the IK solver. Hence, a large set of promising seeds are available which all represent configurations of the end effector that lie in the surrounding of $\mathbf{p}$. Form these poses the IK solver has a good chance to find a solution without getting stuck in local minima or being affected by singularities. The method $getNextNeighbor$ of Algorithm 3 starts with returning direct neighbors of the cell that corresponds to $\mathbf{p}$. If these neighbors do not result in an IK solution, higher grade neighbors are generated.

Algorithm 3 ComputeIKExtended($\mathbf{p}$)

Require: IKMap $\mathcal{I}$, IK-Solver $\mathcal{IK}$

```
q ← ComputeIK(p)
if (q) then
    return q
while (!TimeOut() && !MaxCellsProcessed()) do
    p_neighbor ← getNextNeighbor(p)
    v ← I.getGridCell(p_neighbor)
    q_seed ← v.getReferenceConfig()
    IK.setSeed(q_seed)
    q ← IK.solve(p)
    if (q && CollisionFree(q)) then
        return q
return NULL
```

IV. EVALUATION

We evaluate the *IK–Map* approach with a kinematically simulated model of the humanoid robot ARMAR-4 with the robotics simulation toolbox Simox[1] [24]. The kinematic chain used in the following evaluation setups covers 2 hip and 8 arm joints.

[1]http://gitlab.com/Simox

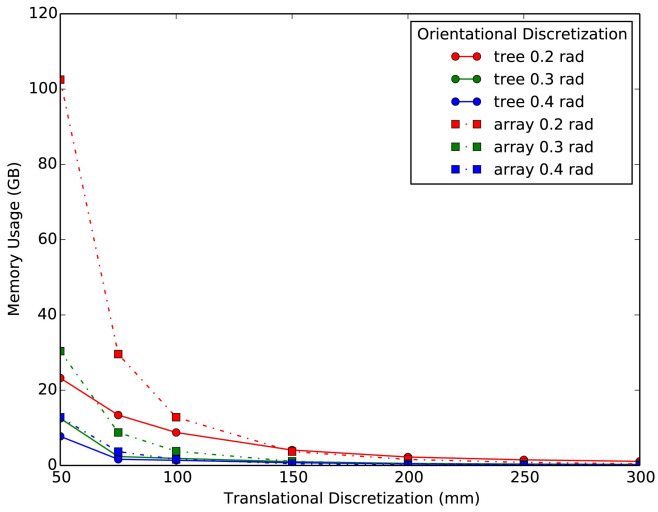

Fig. 3. The effect of different discretization parameters on the memory consumption. The solid lines show the measured results of the proposed *IK-Map* approach, i.e the data is stored in a tree structure. For comparison, the minimal memory consumption that would be needed when storing the data in an array-like structure is shown with dashed lines.

A. Memory Consumption

Although systems tend to be equipped with large amounts of memory, the memory consumption of the *IK-Map* data structure can be challenging. Compared to related approaches, where only one quality value is stored in each grid cell, the *IK-Map* approach additionally stores a reference configuration in each cell. Hence, the memory consumption per cell, including 1 byte for the quality value, increases from 1 to $1 + (4 * n_{dof})$ byte, whereas n_{dof} is the number of DoF and each joint value is stored as 4 byte, representing a floating point value. Depending on the discretization parameters, the resulting memory assignment can be large. In Fig. 3, the effects of different translational (from 50 to 300 mm) and rotational (0.2, 0.3 and 0.4 radian) discretization parameters are shown.

The figure shows results from data structures that have been built from 100 million random samples of the 10 DoF kinematic chain covering 2 torso and 8 arm joints of ARMAR-4. The generation of such data structures takes about 9 hours on a standard Linux PC, utilizing all 8 cores. In addition to the measured memory usage of the *IK–Map* approach (solid lines), the minimal memory consumption for array-like data structures is depicted with dashed lines. The results show, that the discretization parameters influence the memory consumption directly and that storing the data in tree-like data structures is more memory efficient compared to array-based data management.

B. Performance and Quality

A 10 DoF kinematic chain covering torso and one arm of ARMAR-4 was used to analyze the gain in performance and quality when using the *IK–Map* approach. For evaluation purposes the same 10000 IK queries are processed by four different approaches:

1) *NoMap*$_1$: Iterative IK solver similar to the approaches

of [3], [1] that is seeded with one standard configuration.
2) *NoMap*$_{100}$: Iterative IK solver that is seeded with up to 100 random configurations.
3) *IK–Map*$_1$: *IK–Map* approach, without considering neighbors (Algorithm 2).
4) *IK–Map*$_{100}$: *IK–Map* approach that seeds the IK solver with up to 100 reference solutions of the neighboring cells (Algorithm 3).

To ensure that all 10000 IK targets are reachable, joint configurations are randomly generated and forward kinematics is used to build the IK targets in workspace. The full 6D pose has to be calculated by the different approaches.

The *IK–Map* used in this evaluation has been built with discretization parameters of 150 mm (translational part) and 0.3 radian (rotational part). The data structure was calculated by the use of 100 million random samples resulting in a map size of 1.29 GB.

As shown in Table I, the use of a standard iterative IK solver [3], [1] that always starts from a fixed initial configuration (column 1) results in a bad success rate, since the distance from the initial seed to the target pose is large (681 mm and 2.61 radian on average) resulting in bad performance of the gradient descent approach. Using the *IK–Map* approach for seeding the IK solver drastically reduces the distance that the IK algorithm needs to overcome to reach the target pose (90 mm and 0.2 radian on average). Hence, the success rate is much higher (98.9%), the performance is better (1.1 ms per IK query) and the resulting manipulability value is higher (an average value of 0.106 compared to 0.067).

Both approaches take advantage from using multiple starting seeds (column 2: random seeds, column 4: considering neighboring cells with the *IK–Map* approach). By using up to 100 seeds the success rate of the IK query could be increased to up to nearly 100%. Again, the use of the *IK–Map* approach outperforms the standard approach in terms of performance (4.5 ms versus 1.2 ms on average per IK query) and manipulability of the IK results (from 0.069 to 0.106 on average). The effect of having results with better manipulability can be seen in Fig. 4. On the left, an IK solution, generated by the iterative IK approach (2) is shown. The right side shows a solution of the *IK–Map* approach (4) that resulted in a configuration with higher manipulability.

C. Parameter Evaluation

The effect of the different discretization parameters on the success rate can be seen in Fig. 5. The first two columns show the average success rate of the *NoMap*$_1$ and *NoMap*$_{100}$ approaches. The following columns depict how different parametrization of the cell size in translational (50, 150 and 300 mm) and rotational (0.2, 0.3, and 0.4 radian) dimensions influence the achievable success rate of the IK solver. The *IK–Map*$_1$ and *IK–Map*$_{100}$ are shown in blue and red, respectively. It can be seen that there is no effect of different cell sizes for the *IK–Map*$_{100}$ approach; all IK querys have been answered successfully. Further, the performance of the *IK–Map*$_1$ algorithm is weak for small cell sizes. This caused by the fact that due to the randomized buildup approach several cells haven't been filled during the offline procedure. Hence, the IK solver was not be able to find a suitable seed since no neighbors were queried in this approach. This artefact can be avoided through longer buildup procedures, possibly in addition with post processing (e.g. smoothing) operations.

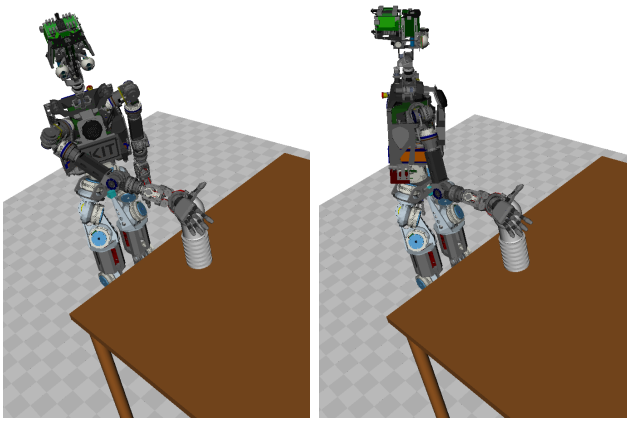

Fig. 4. Exemplary IK solutions. Left: The iterative IK solver resulted in a configuration with low manipulability. Right: The *IK–Map* approach generated a solution with better manipulability.

TABLE I

COMPARISON BETWEEN STANDARD ITERATIVE IK CALCULATIONS AND THE *IK–Map* APPROACH.

	(1) NoMap$_1$	(2) NoMap$^*_{100}$	(3) *IK–Map*$_1$	(4) *IK–Map*$^{**}_{100}$
Success rate	25.64%	99.97%	98.93%	100.00%
Avg. manip.	0.0674	0.0693	0.1059	0.1055
SD. manip.	0.0368	0.0369	0.0368	0.0370
Avg. time IK [ms]	1.6	4.5	1.1	1.2
Avg. dist. pos. [mm]	681.12	662.83	90.36	90.31
SD. pos. [mm]	246.86	269.30	35.47	35.45
Avg. dist. rot. [rad]	2.61	2.39	0.18	0.19
SD. rot. [rad]	0.86	0.95	0.08	0.08

* In case of failure, the IK solver runs up to 100 times with a randomly sampled seed.
** The *IK–Map* solver searches up to 100 neighboring cells for promising seeds.
SD Standard deviation

In Fig. 6 the effect of parameter selection on the IK query time is depicted. In this evaluation, only successful IK querys are taken into account. The first two columns show the average IK query time of the *NoMap*$_1$ and *NoMap*$_{100}$ approaches. The results of the *IK–Map*$_1$ and *IK–Map*$_{100}$ are shown in the following columns in blue and red. It can be seen that a larger cell size results in an increased runtime of the IK algorithm which is caused by the larger distances that the iterative IK approach needs to overcome to find a solution.

V. CONCLUSION

In this work, the *IK–Map* approach has been introduced in order to improve inverse kinematics computation. *IK–Maps* are an extension of the reachability map approach that allows to additionally store reference IK solutions in the workspace data structure. This data helps to speedup

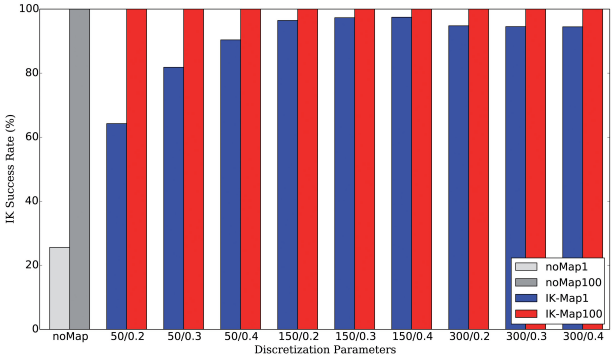

Fig. 5. The effect of the discretization parameters on the IK success rate.

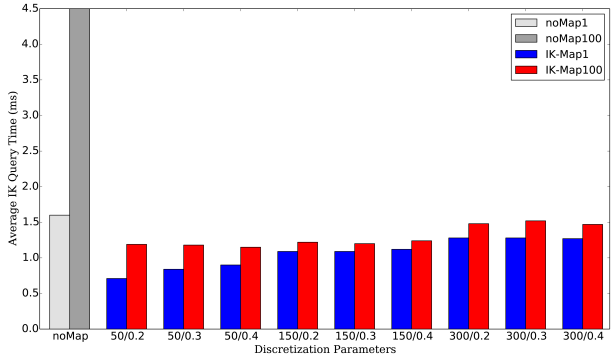

Fig. 6. The effect of different discretization parameters on the IK query time (only successful querys are considered).

iterative IK solvers which are based on gradient descent approaches and hence need a seed to start the IK search. The approach was evaluated with a 10 DoF kinematic chain of the humanoid robot ARMAR-4 by solving a large number of IK queries. The memory consumption as well as the influence of parameter selection was investigated. We showed that the performance as well as the success rate of iterative IK solvers could be increased by using the *IK–Map* approach. Further, it has been evaluated that the quality of the IK results can be improved in terms of manipulability.

Nevertheless, several issues could be addressed in order to improve the proposed approach. First, the offline generation of the data could be improved to receive a better representation of the workspace. With hybrid approaches, the results of the map generation can be post processed in order to counteract the effects of random sampling. In addition, more advanced IK solving algorithms can be used with the *IK–Map* approach to tackle more challenging IK problems, e.g. by considering the stability of the robot or interaction forces with the environment.

REFERENCES

[1] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," University of California, San Diego, Tech. Rep., 2004.

[2] C. W. Wampler, "Manipulator inverse kinematic solutions based on vector formulations and damped least-squares methods," in *IEEE Transactions on Systems, Man and Cybernetics*, vol. 1, 1986, pp. 93–101.

[3] B. Siciliano and J.-J. Slotine, "A general framework for managing multiple tasks in highly redundant robotic systems," in *Advanced Robotics, 1991. 'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, vol. 2, 1991, pp. 1211–1216.

[4] N. Mansard, O. Stasse, P. Evrard, and A. Kheddar, "A versatile generalized inverted kinematics implementation for collaborative working humanoid robots: The stack of tasks," in *Advanced Robotics, 2009. ICAR 2009. International Conference on*, June 2009, pp. 1–6.

[5] A. Escande, N. Mansard, and P. B. Wieber, "Fast resolution of hierarchized inverse kinematics with inequality constraints," in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, 2010, pp. 3733–3738.

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

[7] R. Diankov, "Automated construction of robotic manipulation programs," Ph.D. dissertation, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, September 2010.

[8] O. Porges, T. Stouraitis, C. Borst, and M. Roa, "Reachability and capability analysis for manipulation tasks," in *ROBOT2013: First Iberian Robotics Conference*, ser. Advances in Intelligent Systems and Computing, M. A. Armada, A. Sanfeliu, and M. Ferre, Eds. Springer International Publishing, 2014, vol. 253, pp. 703–718.

[9] A. Aristidou and J. Lasenby, "Inverse kinematics: a review of existing techniques and introduction of a new fast iterative solver," Department of Information Engineering, University of Cambridge, Tech. Rep. CUEDF-INFENG, TR-632, September 2009.

[10] C. Lee and M. Ziegler, "Geometric approach in solving inverse kinematics of puma robots," *Aerospace and Electronic Systems, IEEE Transactions on*, vol. AES-20, no. 6, pp. 695–706, nov. 1984.

[11] T. Asfour and R. Dillmann, "Human-like motion of a humanoid robot arm based on a closed-form solution of the inverse kinematics problem," in *IEEE International Conference on Intelligent Robots and Systems (IROS)*, vol. 2, no. October, 2003, pp. 1407–1412.

[12] A. Balestrino, G. De Maria, and L. Sciavicco, "Robust control of robotic manipulators," in *9th IFAC World Congress*, 1984, pp. 2435–2440.

[13] S. R. Buss and J.-S. Kim, "Robust control of robotic manipulators," *Journal of Graphics Tools*, pp. 37–49, 2005.

[14] A. Escande, N. Mansard, and P.-B. Wieber, "Hierarchical quadratic programming: Fast online humanoid-robot motion generation," *The International Journal of Robotics Research*, pp. 1006–1028, 2014.

[15] M. Kallmann, "Analytical inverse kinematics with body posture control," *Comput. Animat. Virtual Worlds*, vol. 19, pp. 79–91, May 2008.

[16] K. Grochow, S. L. Martin, A. Hertzmann, and Z. Popović, "Style-based inverse kinematics," in *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3. ACM, 2004, pp. 522–531.

[17] I. Ebert-Uphoff and G. Chirikjian, "Inverse kinematics of discretely actuated hyper-redundant manipulators using workspace densities," in *IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, Apr 1996, pp. 139–145.

[18] Y. Wang and G. Chirikjian, "Workspace generation of hyper-redundant manipulators as a diffusion process on se(n)," *Robotics and Automation, IEEE Transactions on*, vol. 20, no. 3, pp. 399–408, June 2004.

[19] N. Vahrenkamp, T. Asfour, and R. Dillmann, "Efficient inverse kinematics computation based on reachability analysis," *International Journal of Humanoid Robotics*, vol. 9, no. 4, pp. 1–25, 2012.

[20] N. Vahrenkamp and T. Asfour, "Representing the robots workspace through constrained manipulability analysis," *Autonomous Robots*, pp. 17–30, 2015.

[21] P. Kaiser, D. Gonzalez-Aguirre, F. Schültje, J. Borràs, N. Vahrenkamp, and T. Asfour, "Extracting whole-body affordances from multimodal exploration," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2014, pp. 1036–1043.

[22] N. Vahrenkamp, T. Asfour, G. Metta, G. Sandini, and R. Dillmann, "Manipulability analysis," in *Proceedings of IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, 2012, pp. 568–573.

[23] T. Asfour, J. Schill, H. Peters, C. Klas, J. Bücker, C. Sander, S. Schulz, A. Kargov, T. Werner, and V. Bartenbach, "ARMAR-4: A 63 DOF Torque Controlled Humanoid Robot," in *IEEE/RAS International Conference on Humanoid Robots (Humanoids)*, 2013, pp. 390–396.

[24] N. Vahrenkamp, M. Kröhnert, S. Ulbrich, T. Asfour, G. Metta, R. Dillmann, and G. Sandini, "Simox: A robotics toolbox for simulation, motion and grasp planning," in *International Conference on Intelligent Autonomous Systems (IAS)*, 2012, pp. 585–594.