# Voxel-Based Motion Bounding and Workspace Estimation for Robotic Manipulators

Peter Anderson-Sprecher and Reid Simmons

***Abstract*— Identification of regions in space that a robotic manipulator can reach in a given amount of time is important for many applications, such as safety monitoring of industrial manipulators and trajectory and task planning. However, due to the high-dimensional configuration space of many robots, reasoning about possible physical motion is often intractable.**

**In this paper, we propose a novel method for creating a *reachability grid*, a voxel-based representation that estimates the minimum time needed for a manipulator to reach any physical location within its workspace. We use up to second-degree constraints on joint motion to model motion limits for each joint independently, followed by successive voxel approximations to map these limits on to the robot's physical workspace. Results using a simulated manipulator indicate that our method can produce accurate reachability grids in real-time, even for robots with many degrees of freedom. Furthermore, errors are almost exclusively biased towards producing more optimistic reachability estimates, which is a desirable characteristic for many applications.**

## I. INTRODUCTION

Modeling the motion of robotic manipulators is an open problem in robotics. Globally, manipulators are constrained to their workspace, the set of all reachable regions, and within a workspace, some regions may be reached quickly, whereas others require a longer time. Knowing which regions are reachable in a short amount of time, which are more distant, and which are wholly unreachable is useful both to the robot, which must plan its movements according to its limitations, and to outside agents, which may need to take possible robot motions into account when planning their own movements. However, finding these regions is made difficult by the high-dimensional configuration space of most manipulators.

A key application is manipulator safety monitoring in unconstrained environments, which requires a conservative estimation of regions into which a manipulator may enter in the near future in order to re-adjust or stop motion whenever humans might enter the robot's workspace [3]. Other possible applications include trajectory and task planning, as manipulators can use motion bounds to prioritize closer objectives over farther ones, or use the estimated time as a distance heuristic in motion planning.

How much time is needed for a robot to reach a given position depends on constraints on its joint motion, usually enforced by the controller based on mechanical and power limitations. Constraints usually take the form of limited joint positions, velocities, and/or accelerations, though higher-order terms such as jerk limits are sometimes used. These constraints allow reasoning about how much time is required to reach a given joint pose given an initial position and velocity. Figure 2a shows an example map of reachability times in joint space, using constant joint velocity limits.

However, targets and obstacles are represented in physical (metric) space rather than joint space. This means that to have useful reachability information, we need to compute the robot's workspace and map minimum reachable times for corresponding joint values into it. Figure 2b shows

This work was supported by the Office of Naval Research, USA.

Peter Anderson-Sprecher and Reid Simmons are with the Robitics Institute at Carnegie Mellon University, Pittsburgh, PA 15213 USA {psprecher@gmail.com, reids@cs.cmu.edu}

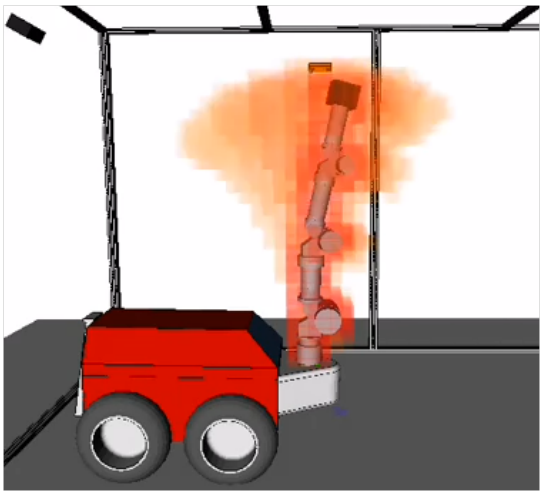


Fig. 1. Reachability grid for a 7 degree-of-freedom manipulator. Orange regions require more time to reach than red regions.

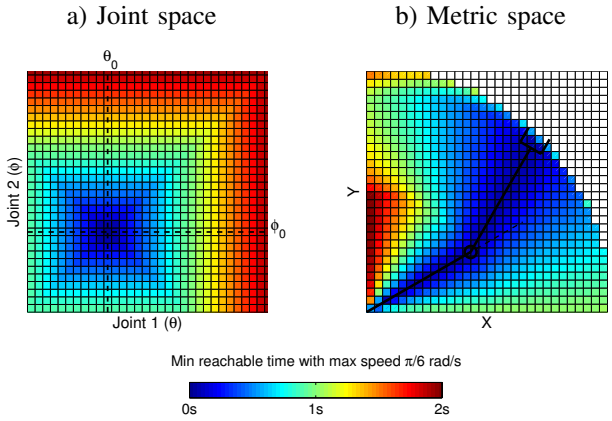



Fig. 2. Reachability grid illustration for 2 degree-of-freedom planar arm. a) Reachability in joint space. b) Reachability in metric space. Arm is drawn in current pose $(\theta_0, \phi_0)$.

the example robot's workspace with associated minimum reachable times. While this is straightforward for a simple robot such as this, real robots often have joint spaces that are too high-dimensional to exhaustively search all possible poses. For example, Figure 1 shows a reachability grid for a 7 degree-of-freedom (DoF) arm. While the physical space swept out by the arm is small, the corresponding 7-dimensional joint space is too large to work with directly.

We solve this problem by collapsing the swept volume of each link to an intermediate voxel approximation before considering it in the computation of previous links, which allows us to avoid searching the entire high-dimensional joint space and instead calculate bounds for each joint independently. The drawback is that it may produce optimistic times in some cases, as it cannot reason about self-collisions and obstacles. However, the general solution is intractable for high-DoF manipulators, and given that approximations are required, optimism is desirable for many applications, such as robot safety monitoring and A* planning heuristics.

Depending on the application, we may care about measuring regions that are reachable by any part of the manipulator (e.g., safety), or regions reachable by just the end effector (e.g., planning). Our experiments focus on the safety application, measuring reachability with respect to the entire manipulator, but results would apply equally to modeling the position of the end effector.

In this paper, we present a novel algorithm for producing a *reachability grid,* a voxel grid containing, for each cell in a robot's workspace, the minimum amount of time needed to reach it. Our approach consists of two parts: first, we produce an invertible joint motion model that uses second-order constraints to determine time bounds for each individual joint. Second, we map these models into a robot's workspace using a recursive voxel-based workspace approximation.

Since this is an approximate algorithm, we empirically measure the accuracy and computational speed of the approach on a simulated 4 DoF manipulator. Our results indicate that accurate reachability grids can be computed in real-time, even for robots with many degrees of freedom, and that approximation errors are minor and almost exclusively biased towards optimistic reachability estimates.

### A. Prior work

Zacharias et al [9], [11] have used reachability grids to model possible robot motion and object manipulability. Their approach has the advantage of considering end effector orientation as well as position, but it cannot operate in real time, and assumes an analytical solution for the inverse kinematics of the manipulator. Others have used voxelization for estimation of swept volume, similar to how we compute the volume swept by a robot link [1].

There has also been significant work in robot workspace estimation, a key component of our work. Several works deal with workspace solutions tailored to specific manipulators [5], [8]. Others have explored more general analytical approaches [2], [6], [10]. Rastegar and Deravi studied the effect of joint motion constraints on workspaces [7].

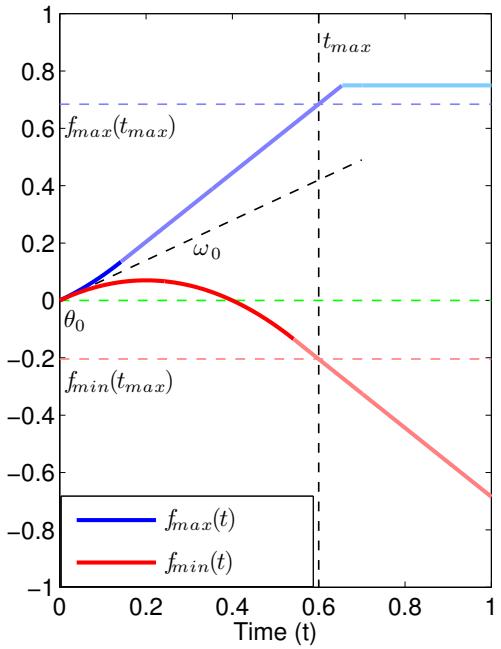



Fig. 3. Maximum motion bound $f_{max}(t)$ and minimum motion bound $f_{min}(t)$ (Eq 3) for a single manipulator joint. Joint has initial position $\theta_0$ and initial angular velocity $\omega_0$. Bounds increase quadratically until velocity limit is reached, at which point bounds increase linearly until the final position limit.

However, none of these approaches operate in real time, and application of any workspace-based method would have to be run many times on varying sets of joint constraints to give a full reachability grid.

## II. Bounding Joint Motion

In order to bound the time needed to reach any given joint position, we first require a set of *a priori* motion constraints for each joint. These constraints may consist of:

1) Constrained joint angle $\theta \in [\theta_{min}, \theta_{max}]$
2) Constrained joint velocity $\omega \in [\omega_{min}, \omega_{max}]$
3) Constrained joint acceleration $\alpha \in [\alpha_{min}, \alpha_{max}]$

Any subset of these constraints may be used; absent constraints are assumed to be infinite. In principle, we could also use other constraints, such as jerk limits, but these are not analytically invertible so it is preferable to avoid them, if possible.

Using the above constraints and a given initial joint position $\theta_0$ and velocity $\omega_0$, we can use piecewise functions to place bounds $[f_{min}(t),\ f_{max}(t)]$ on the joint motion as a function of $t$. Upper and lower trajectory bounds for a sample joint are shown in Figure 3.

As long as the joint velocity $\omega$ is below the maximum $\omega_{max}$, we assume maximum acceleration and the upper bound is governed by the trajectory quadratic

$$\theta_0 + \omega_0 t + \frac{1}{2}\alpha_{max} t^2. \tag{1}$$

This holds for all $t \leq t_1$ where $t_1$ is the time at which $\omega = \omega_{max}$:

$$t_1 = \frac{\omega_{max} - \omega_0}{\alpha_{max}} \tag{2}$$

For $t > t_1$, $\theta$ rises linearly with slope $\omega_{max}$ until $\theta_{max}$ is reached at time $t_2$ (unless $\theta_{max}$ is reached prior to $t_1$, in which case this linear segment does not exist). This gives the piecewise maximum bound function $f_{max}(t)$:

$$f_{max}(t) = \begin{cases} \theta_0 + \omega_0 t + \frac{1}{2}\alpha_{max} t^2 & \text{if } t \leq \min(t_1, t_2) \\ f_{max}(t_1) + \omega_{max}(t - t_1) & \text{if } t_1 < t \leq t_2 \\ \theta_{max} & \text{otherwise} \end{cases}. \tag{3}$$

The minimum bound $f_{min}(t)$ is derived similarly using $(\theta_{min}, \omega_{min}, \alpha_{min})$.

These bounds provide, for some given maximum time bound $t_{max}$, a corresponding joint range $[f_{min}(t_{max}), f_{max}(t_{max})]$ over which to compute reachability. In practice, $t_{max}$ is set to an application-specific value corresponding to the maximum useful time horizon. To compute the entire workspace we can let $t_{max} \to \infty$, in which case the range is $[\theta_{min}, \theta_{max}]$.

Finally, in order to compute the minimum reachable time for some joint configuration, we also need to compute the inverse $t = f^{-1}(\theta)$, i.e., given a joint value, find the minimum time needed to reach it.

As long as we use at most second-order constraints, the bounds are analytically invertible. In case of multiple solutions, we take the lowest non-negative time:

$$f_{max}^{-1}(\theta) \quad = \quad \{\min t \,|\, t \geq 0,\, f_{max}(t) = \theta\} \tag{4}$$

The true minimum reachable time for a joint angle is then given by the lesser of the min or the max bound:

$$f^{-1}(\theta) = \min(f_{max}^{-1}(\theta), f_{min}^{-1}(\theta)) \tag{5}$$

Since we assume constraints for different joints are independent, any robot pose now has a minimum reachable time equal to the maximum value of $f^{-1}(\theta)$ across all joints. In the next section, we present a workspace estimation algorithm that uses these bounds to produce a reachability grid in metric space.

## III. Workspace Estimation

The straightforward brute-force approach to mapping joint poses into a workspace would be to sample the entire joint space at regular intervals and save the minimum time value found for each voxel. However, as the size of the joint space is exponential in the number of joints, this method is intractable for high-DoF manipulators.

We instead present an algorithm that uses successive voxel approximations to sweep each joint independently across its range of motion $[f_{min}(t_{max}), f_{max}(t_{max})]$. Our approach is based on the insight that, once we have the workspace for one link calculated in its reference frame, we can calculate the workspace of the previous link recursively by treating the last link's workspace as an attached rigid body. Accordingly, we

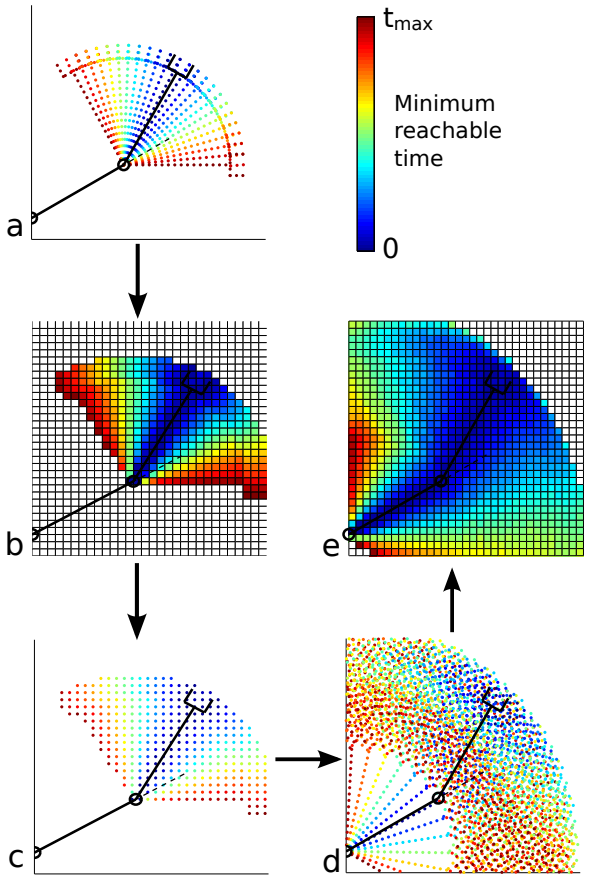



Fig. 4. Workspace reachability grid computation. a) Point-cloud representation of last link is swept through its range of motion. b) Swept cloud is voxelized. c) Reduced cloud extracted from voxel grid. d) First link is added to reduced cloud and swept through its range of motion. e) Final reachability grid is obtained by voxelizing point cloud with minimum time in each voxel.

can use roughly the following steps to compute a reachability grid:

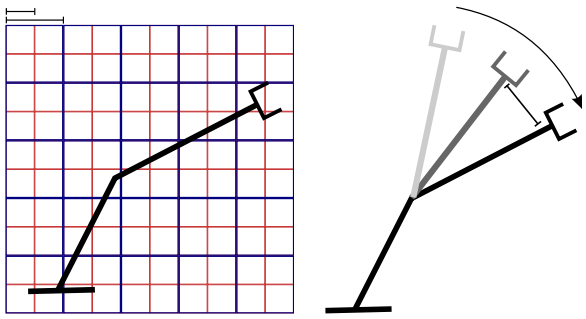
1) Sweep point-cloud representation of last link (end effector) through the range of motion of the last joint
2) Reduce swept cloud by collapsing to intermediate voxel grid
3) Attach reduced point cloud as rigid body to end of link $n-1$ and repeat
4) After all links are swept, voxel grid for base link is final reachability grid

Figure 4 illustrates these steps, and pseudocode is shown in Algorithm 1. All intermediate point clouds and voxel grids must store the estimated minimum reachable time $t$ associated with each point $p$ and voxel $v$, respectively.

Note that the algorithm assumes known point cloud representations of all manipulator links. In practice, a suitable cloud can be produced from a triangle mesh model by sampling the enclosed volume at a density equal to, or greater than, that of the output reachability grid. If reachability is being calculated for only the end effector instead of the entire manipulator body, then no initial point cloud representation is needed and a single point at the end effector will suffice.

**Algorithm 1** Reachability grid workspace computation

```
cloud ← ∅
finalGrid ← fill(∞)
for link = n to 1 do
  cloud.addPoints(cloud(link), 0) {t = 0 at θ = θ_0}
  subGrid ← fill(∞)
  for θ = f_min(t_max) to f_max(t_max) step Δθ do {See eq. 3,6}
    for all (p, t) in cloud do
      p' ← rotate(p, θ)
      t' ← max(t, f^{-1}(θ)) {See eq. 5}
      if link > 1 then
        v ← subGrid.getVoxelByPoint(p')
      else
        v ← finalGrid.getVoxelByPoint(p')
      end if
      v.t ← min(v.t, t')
    end for
  end for
  if link > 1 then
    cloud ← voxelCenters(subGrid)
  end if
end for
return finalGrid
```



a) Subvoxelization ratio b) Step factor

Fig. 5. Parameters used in workspace estimation. a) Subvoxelization ratio is of the voxel size used in cloud reduction (red) to that of the final reachability grid (blue). Ratio shown is 1/2. b) Step factor is maximum point motion between steps of $\theta$ during sweep, as a fraction of final voxel size. Factor shown is $\approx 1$.

The point cloud reduction at each joint allows this to run efficiently, even on high DoF manipulators for which an exhaustive search would be impossible. If all points were kept at each stage, the number of points in the cloud would increase exponentially with each manipulator link, making the computation intractable beyond a few degrees of freedom. Given a robot with $n$ degrees of freedom, each of which has $\psi$ angular range of motion, our grid reduction algorithm runs in linear time, $O(n\psi)$, whereas a brute-force solution requires exponential time, $O(\psi^n)$.

### A. Approximation errors

The approximation speedup does come at the cost of induced error, as every time we collapse a point cloud into an intermediate voxel grid, every point will move to the nearest voxel center. To minimize this effect, we typically want to use a smaller grid for the intermediate grid (i.e., $subGrid$ in Algorithm 1) than the final reachability grid ($finalGrid$ in Algorithm 1). As we shrink the intermediate voxel size, we can approach zero error, but in doing so point clouds reduce less and performance approaches that of brute-force.

An additional source of error is in sweeping through the possible angles for each joint. We have to choose how finely to sample these angles; coarser sampling will mean faster computation and smaller clouds, but at the cost of possible skipped regions.

To control these factors, we introduce two parameters, as illustrated in Figure 5, that we can use to trade off between accuracy and computational speed. Both parameters are normalized to the final reachability grid cell size.

1) The **subvoxelization ratio** is the ratio of the voxel size of the intermediate grids to the voxel size of the final grid.
2) The **step factor** bounds the step size used when sweeping the point cloud through a joint's range of motion.

The step parameter $\Delta\theta$ is set based on the step factor $s$ such as to limit motion of all points $p$ around the joint axis $a$ to some fraction of the voxel size $v$:

$$\Delta\theta = \frac{sv}{\max(a \times p)} \tag{6}$$

We expect the subvoxelization ratio to have a significant effect on error, much more so than the step factor. Voxelization can shift points at most half the diagonal length of a voxel for each link, so we get a maximum displacement error bound $\epsilon$ for an $n$-DoF manipulator and a subvoxel length $l$ of $\epsilon \leq \frac{\sqrt{3}}{2} nl$. However, in practice the error will be much less than this. In the next section, we examine the error rate empirically, and test the effect of our approximation parameters on accuracy and computation speed.

## IV. Experimental methods

All tests used a simulated 4-DoF manipulator with two pitch and two roll joints, as shown in Figure 6. This arm was chosen because computing ground truth for a higher-DoF arm is prohibitively slow. The arm is 1m long at full extension, and the calculated reachability grid has a voxel width of 5cm. All computation was single-threaded and executed on an Intel Core 2 Quad desktop computer with 2GB of RAM. For robot simulation and visualization, we used the OpenRAVE simulator [4].

Reachability grids were gathered at 10 randomly selected poses. Joint constraints consisted of a maximum speed of $|\omega| \leq 1$rad/s. There were no position or acceleration constraints. Reachability was computed out to $t_{max} = 0.5s$, giving a total freedom of 1 radian in each joint.

For each test pose, a ground truth reachability grid was computed using a uniformly sampled brute-force search of the joint space. The step factor when calculating ground truth

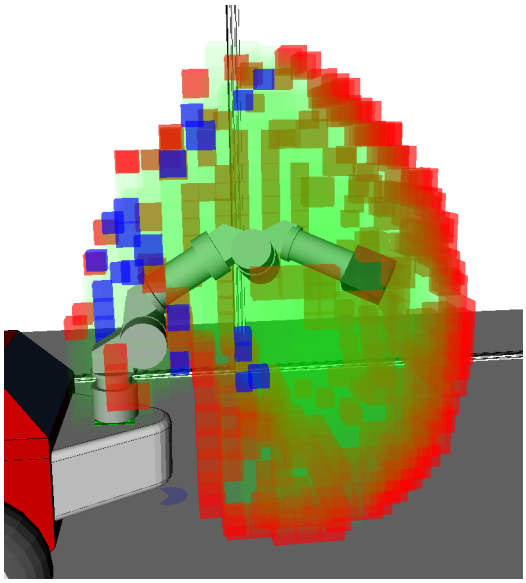


Fig. 6. Test 4-DoF manipulator with observed errors. Red voxels are false positives (present only in estimated grid), blue voxels are false negatives (present only in ground truth), and the green voxels are correct (present in both estimated and ground truth). Parameters used to generate this image are 0.5 for both subvoxelization ratio and step factor.

grids was 0.4, ensuring a worst-case distance between sweep steps of 2cm.

Comparison grids were generated with subvoxelization ratios ranging from 0.1 to 1.0, and step factors ranging from 0.1 to 1.5, both at 0.025 increments. False positives and negatives were measured by comparing the full measured workspace against the ground truth grid, i.e., we compared the set of all voxels with minimum reachable time $t \leq 0.5s$.

## V. Experimental Results

Results indicate that our approach does produce an accurate workspace estimate, particularly at low subvoxelization ratios. Figure 6 shows an example of errors observed for our 4-DoF test arm. Figure 7 shows precision, recall, and processing time as a function of the subvoxelization ratio and step factor parameters, averaged over all 10 test poses.

The primary effect of the approximation errors appears to be a slight dilation of the robot workspace, particularly near the end of the manipulator chain. As shown in Figure 6, false positives are more common at the end, while false negatives are uncommon and present only near the base. This is unsurprising, as the large number of points approximated at each step will naturally cause a diffusion effect, which is pronounced with more successive approximations. This means that areas near the end effector will see the most growth due to being moved at every step in the kinematic chain. For most subvoxelization ratios, false positives were no farther than 1 voxel (5cm) from their true locations, but for ratios above about 0.8 some were as distant as 2 voxels. In the reachability grid, the net effect of this dilation is a consistent underestimation of reachability times, the magnitude of which is dependent on the size of the dilation.

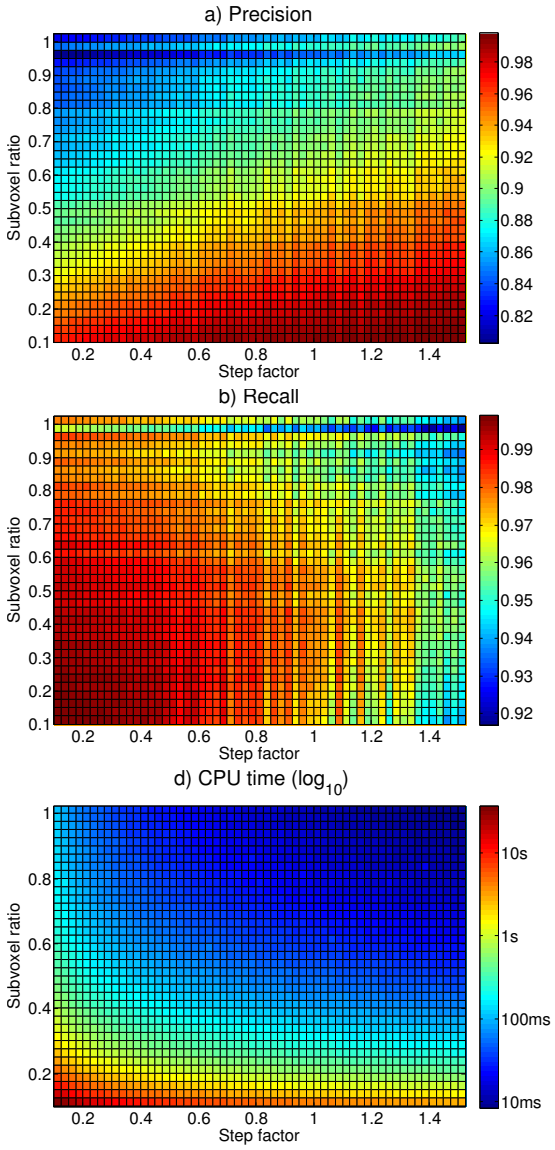



Fig. 7. Experimental results for reachability grid computation. a) Precision, b) Recall, and c) Log CPU processing time, as a function of subvoxelization ratio and step factor parameters (see Figure 5).

As expected, when varying parameters, the subvoxelization ratio has the most significant effect on overall performance. Errors are markedly lower for small ratios, but at the cost of dramatically longer processing times, as the number of subvoxels in the workspace is proportional to the cube of their size. In practice, we found that using a ratio of about a half voxel gave a good trade-off between performance and accuracy.

As is shown in Figure 7(a,b) certain structured error appears at subvoxelization ratios close to, but under, 1, most noticeably at 0.975. This is due to the relationship between the voxel positions in the intermediate grid and the final grid. When there is a small difference in size between the intermediate and final voxels, the relative positions of voxel centers will be consistently biased over large regions of space, meaning that, depending on the position of the

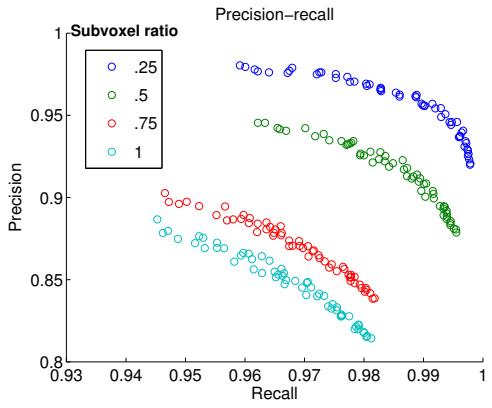



Fig. 8. Precision-recall curves generated by varying step factor at selected subvoxelization ratios.

manipulator workspace, boundaries may grow or shrink considerably. In practice, this means that subvoxelization ratios close, but not equal, to 1 should be avoided, if possible.

Varying the step factor does not have a clear-cut effect on overall accuracy, but lower values appear to produce a more generous workspace estimate, whereas higher values produce a more conservative estimate. This is because lower ratios produce larger point clouds, which will naturally diffuse to create larger workspaces. Precision-recall curves generated by varying the step factor are shown in Figure 8. Step factors significantly above 1 run the risk of skipping over regions and may cause gaps in the swept volume. However, due to the cloud diffusion effect and the fact that the step factor is a worst-case distance, we did not observe this below factors of about 1.25. The step factor does not have a major effect on speed unless very small values are used (computation time should vary with its reciprocal). We found that a step factor of about 1 gave a good trade-off between precision and recall.

For robots with higher degrees of freedom than our test robot, we would expect the maximum displacement error, and correspondingly the false positive rate, to increase linearly with the number of degrees of freedom. However, this is difficult to verify empirically due to the intractability of computing ground truth reachability for high-DoF robots. That said, we have used our approach on a 7-DoF arm (as shown in Figure 1) and results appear consistent with those seen at the 4-DoF level. The video accompanying this paper demonstrates reachability for the 7-DoF arm as it sweeps through a series of motions, as computed in real-time at 10hz.

## VI. Conclusions and Future Work

Our results indicate that *reachability grids* are a fast and accurate way to compute motion bounds, even for high degree-of-freedom manipulators. Errors typically result in a displacement of no more than a few voxels away from true locations. The algorithm can run in real-time for even high-dimensional manipulators, is easily parallelizable, and many of the operations could be implemented on graphics hardware for further speedups.

The fact that our approach does not take collisions into account, combined with approximations in the workspace estimation, often results in underestimates of reachability times but rarely or never results in overestimates. This makes it particularly well-suited to applications that require an optimistic estimate of reachability, such as robot safety monitoring and A* path planning heuristics.

While positional bounding as we have demonstrated here is sufficient for some applications, many manipulation tasks need to take end effector orientation into account as well. In principle our approach should work in a higher-dimensional space, but performance will be reduced significantly, so further study is needed to demonstrate its suitability.

Another key limitation of our approach is that it is unable to take any collisions or obstacles into account. Opportunities for improvement here may be limited, as fully exploring the joint space is inherently intractable for high-DoF manipulators. However, with further research it may be possible to reason about collisions in a limited fashion by using the existing method to carve out large regions of obstacle-free joint space while spending extra time in areas near obstacles.

## VII. Acknowledgements

Thanks to Daniel Huber, Paul Rybski, Chris Niessl, and Thomas Koletschka for their work on the Intelligent Monitoring of Assembly Operations project. This work was supported by the Office of Naval Research under contract #N00014-09-D-0584.

## References

[1] K. Abdel-Malek, D. Blackmore, and J. Yang. Swept volumes: Foundations, perspectives, and applications. *International Journal of Shape Modeling*, 12:87–127, 2006.

[2] K. Abdel-Malek, H.J. Yeh, and N. Khairallah. Workspace, void, and volume determination of the general 5dof manipulator. *Mechanics of Structures and Machines*, 27:91–117, 1999.

[3] P. Anderson-Sprecher. Intelligent monitoring of assembly operations. Technical Report CMU-RI-TR-02-03, Robotics Institute, Carnegie Mellon University, June 2011.

[4] R. Diankov. *Automated Construction of Robotic Manipulation Programs*. PhD thesis, Carnegie Mellon University, Robotics Institute, August 2010.

[5] F. Freudenstein and E.J. Primrose. On the analysis and synthesis of the workspace of a three-link, turning-pair connected robot arm. *Journal of Mechanisms, Transmissions, and Automation in Design*, 106:365–370, 1984.

[6] E. J. Haug, Chi-Mei Luh, F. A. Adkins, and Jia-Yi Wang. Numerical algorithms for mapping boundaries of manipulator workspaces. *Journal of Mechanical Design*, 118(2):228–234, 1996.

[7] J. Rastegar and P. Deravi. The effect of joint motion constraints on the workspace and number of configurations of manipulators. *Mechanism and Machine Theory*, 22(5):401 – 409, 1987.

[8] J. Spanos and D. Kohli. Workspace analysis of regional structures of manipulators. *J. Mech. Transm. Autom. Des.*, 107(2):216–222, Jun 1985.

[9] F. Stulp, A. Fedrizzi, F. Zacharias, M. Tenorth, J. Bandouch, and M. Beetz. Combining analysis, imitation, and experience-based learning to acquire a concept of reachability. In *9th IEEE-RAS International Conference on Humanoid Robots*, pages 161–167, 2009.

[10] Y. C. Tsai and A. H. Soni. An algorithm for the workspace of a general n-r robot. *ASME Journal of Mechanisms, Transmissions, and Automation in Design*, 105:52–57, 1983.

[11] F. Zacharias, C. Borst, and G. Hirzinger. Capturing robot workspace structure: representing robot capabilities. In *IROS*, pages 3229–3236, 2007.