Numerical Convolution on the Euclidean Group with Applications to Workspace Generation

Gregory S. Chirikjian, *Member, IEEE,* and Imme Ebert-Uphoff

***Abstract*—In this work, the concept of a convolution product of real-valued functions on the Special Euclidean group, $SE(D)$ (which describes all rigid body motions in D-dimensional Euclidean space), is applied to the determination of workspaces of discretely actuated manipulators. These manipulators have a finite number of joint states. If a discretely actuated manipulator consists of P actuated modules, each with K states, then it can reach K^P frames in space. Given this exponential growth in the number of reachable frames, brute force representation of discretely actuated manipulator workspaces is not feasible in the highly actuated case. However, by partitioning a discretely actuated manipulator into P modules, and approximating the workspace of each module as a density function on a compact subset of the Special Euclidean group, the whole workspace can be approximated as an P-fold convolution of these densities. A numerical approximation of this convolution is presented in this paper which is $\mathcal{O}(P)$ for fixed taskspace dimension. In the special case when the manipulator is composed of P identical actuated modules, the workspace density for the whole manipulator can be calculated in $\mathcal{O}(\log P)$ computation time. In either case, the $\mathcal{O}(K^P)$ computations required by brute force workspace generation are avoided.**

***Index Terms*— Convolution, discrete actuation, Euclidean group, manipulator workspaces, rigid body motion.**

I. Introduction

THE CONCEPT of a convolution product of real-valued functions on the Special Euclidean Group (which describes rigid-body motion in Euclidean space) is used in this paper. The primary application is the generation of discretely actuated manipulator workspaces, and determination of the density of reachable frames in any portion of the workspace. While a number of works address the problem of determining workspaces, the concept of density is not usually addressed. In the context of discrete actuation, the density of frames (number of frames per unit taskspace volume[1]) in many ways replaces dexterity measures as a scalar function of importance defined over the workspace. This is because density in the neighborhood of a given point/frame is an indication of how accurately a discretely actuated manipulator can reach that point/frame.

Manuscript received Dec. 8, 1995; revised November 15, 1996. This work was sponsored by National Science Foundation Grants IRI-9357738 and IRI-9453373. This paper was recommended for publication by Associate Editor J. Wen and Editor S. Salcudean upon evaluation of the reviewers' comments.

The authors are with the Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD 21218 USA.

Publisher Item Identifier S 1042-296X(98)00598-9.

[1]The taskspace for positioning and orienting an object in three dimensional space is six-dimensional.

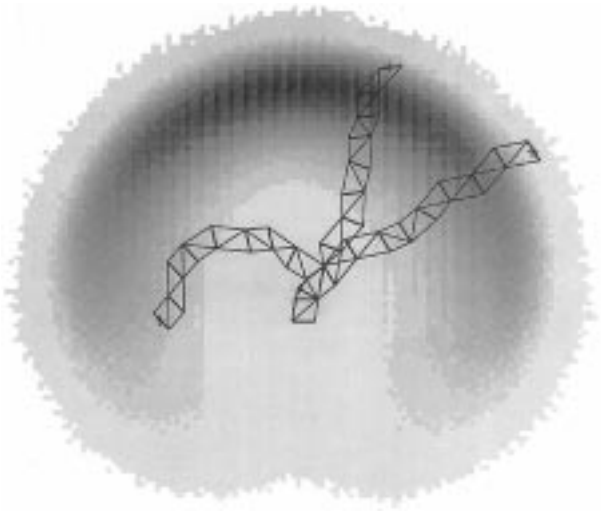

Fig. 1. A discretely actuated manipulator with 4^{30} States.

To compute this workspace density function using brute force is computationally intractable, e.g., it requires $\mathcal{O}(K^P)$ evaluations of the forward kinematic equations for a manipulator with P actuated modules each with K states. In addition, an array storing the density of all volume elements in the workspace must be incremented $\mathcal{O}(K^P)$ times if brute force computation is used.

Fig. 1 shows a schematic of the density of frames reachable by a discretely actuated variable geometry truss manipulator. If there are 30 actuated truss elements (ten modules) and each element has four states (and thus the whole manipulator has $4^{30} \approx 10^{18}$ states) the workspace density cannot simply be computed using brute force because this would take hundreds of years using current computer technology. This combinatorial explosion is a major reason why discrete actuation is not commonly used, despite the fact that the concept is almost three decades old (see e.g., [22], [25]). Having a representation of the density of reachable frames is important for performing inverse kinematics and design of discretely actuated manipulators. Using the concept of convolution discussed in the sequel, an approximation of the workspace density can be achieved in $\mathcal{O}(\log P)$ calculations for macroscopically serial manipulators[2] composed of P identical modules. This reduces the computation time to minutes.

The remainder of this paper is organized as follows. We begin by reviewing relevant literature in Section II. In

[2]A serial cascade of modules where each module may be a serial or parallel kinematic structure.

Section III, terminology from the pure mathematics literature dealing with Lie groups is reviewed, and the concept of convolution of functions on groups is explained in detail. In Section IV, we show why this concept is important for workspace generation of discretely actuated manipulators. In Section V, the numerical implementation of convolution is explained in the planar case, and the spatial case is discussed. Numerical results are generated in Section VI. Section VII offers some conclusions and describes future work.

II. Literature Review

In this section, the robotics literature which addresses swept volumes and manipulator workspace generation is reviewed.

Sen and Mruthyunjaya [27] use manipulators with only discrete joint states to model continuous-range-of-motion manipulators with limited joint resolution. The results are used to examine the positional accuracy of continuous-range-of-motion robots. In addition, an algorithm is developed to use these results to improve positional accuracy and motion planning for manipulators with only few degrees of freedom. Kumar and Waldron [13] describe a model that relates the joint accuracy to the accuracy of the end effector.

Blackmore and Leu [3] examine the volume swept by a continuous-range-of-motion manipulator changing its configuration. That work also develops the mathematical framework to analyze the volume swept by a manipulator arm. It can also be used to generate the work envelope (i.e., the set of all points that the manipulator can occupy).

The workspaces of manipulators with simple geometric structure can be derived using classical geometric approaches. These methods cover a wide range of special manipulator structures, but they are not applicable for manipulators of general structure. In particular there exist algorithms for the generation of manipulators with an arbitrary number of revolute joints (e.g., Korein [12], Kwon *et al.* [14]).

Another approach solves the problem for manipulators with any kind of joints using the Monte-Carlo method. This approach generates a large number of random actuator values (joint angles), and calculates the corresponding end effector positions (e.g., Alciatore *et al.* [1]). An approximation of the workspace boundaries is found by tracking the border of this point set. In principle the Monte Carlo method is not restricted to manipulators with few degrees of freedom. However, in practice the results lose reliability with increasing degrees of freedom. More specifically for discretely actuated manipulators, areas in the workspaces with very low (but nonzero) density are not picked up by the Monte-Carlo method when the variation in density over they workspace is high.

Rastegar and Deravi [23], [24] consider workspace generation for a manipulator with general structure and n joints. Those authors use an approach similar to the one presented in this paper-dividing the manipulator into parts, sub-workspaces of which are calculated. Other works by those authors include the effect of joint motion constraints on the workspace.

When one considers discretely actuated/binary manipulators with low resolution the workspace can be generated using brute force enumeration of configurations. Furthermore, the inverse problem of designing a binary manipulator to reach a set of specified end effector locations can be achieved as in [4], [5]. However, when one considers situations with a very fine resolution or large number of binary actuators, both the forward problem of workspace determination and the inverse problem of manipulator design cannot be performed in this way.

In addition to workspace shape, the density of points in the workspace of a discretely actuated manipulator is a very important quantity for inverse kinematics and path planning in the case of a large number of discrete actuator states [9]. Previously, we presented a method to efficiently generate workspaces [7] and work envelopes [8] in terms of densities. Those works are based on concatenation of the densities of individual modules by sweeping. This paper reformulates the approach in [7] in the more general context of convolution of functions on Lie groups.

III. The Generalized Convolution Concept

In this section we discuss convolution of functions on Lie groups. We start with an introduction to the concept of convolution of functions on Lie groups in the first sub-section, and apply this definition of convolution to the Euclidean group in the remaining two sub-sections.

A. Convolution of Functions on Lie Groups

In this subsection, a generalization of the concept of convolution to functions on Lie groups is considered. It is assumed that the reader is familiar with the concepts of groups and topological spaces. Readers unfamiliar with these concepts are referred to [26], [30].

Recall that the convolution of square-integrable functions on the real line is defined as the following integral:

$$(\alpha * \beta)(x) = \int_{-\infty}^{\infty} \alpha(\xi)\beta(x-\xi)\,d\xi. \tag{1}$$

To discuss the generalization of this integral to functions on Lie groups, an appropriate volume form/integration measure must be used. We denote a differential volume element at a group element g in a Lie group G as $d\mu(g)$. The calculation of this element for the case of the Euclidean group is explained in the Appendix. The concept of convolution can be generalized to square-integrable functions on arbitrary Lie groups as (see [28], [29])

$$(\alpha * \beta)(g_x) = \int_G \alpha(g_\xi)\beta(g_\xi^{-1} \circ g_x)\,d\mu(g_\xi) \tag{2}$$

where the integral is taken over the whole group, and $g_x, g_\xi \in G$ are arbitrary elements.

To show that this generalized definition is consistent with the convolution defined in (1) we consider the set of real numbers, $\mathbb{R}$, as a group (with scalar addition as group operation). For any element $g_x = x$ of this group the inverse element is

$g_x^{-1} = -x$ and the volume element simplifies to $d\mu(g_\xi) = d\xi$, resulting in the desired equation:

$$\begin{aligned}(\alpha * \beta)(x) &= \int_G \alpha(g_\xi)\beta(g_\xi^{-1} \circ g_x)\, d\mu(g_\xi) \\ &= \int_{\mathbb{R}} \alpha(\xi)\beta(-\xi + x)\, d\xi \\ &= \int_{-\infty}^{+\infty} \alpha(\xi)\beta(x - \xi)\, d\xi. \end{aligned} \tag{3}$$

For a real-valued square-integrable function on an arbitrary Lie group, $\beta(g_x) \in \mathcal{L}^2(G), \beta(g_\xi^{-1} \circ g_x)$ is called a *translation* in the same sense that $\beta(x - \xi)$ is a translation of a function in $\mathbb{R}$. In particular, $\beta(g_\xi^{-1} \circ g_x)$ is called a *left translation*, while $\beta(g_x \circ g_\xi^{-1})$ is called a *right translation*.[3] This does not indicate a direction of motion, but rather the order in which the elements appear. In fact, for $\mathbb{R}$ left and right translations are the same because the group operation is commutative.

Left and right translations have no effect on the volume element in $\mathbb{R}$. That is, for any fixed element $g_x = x \in \mathbb{R}, d\mu(g_x \circ g_\xi) = d\mu(g_\xi \circ g_x) = d\mu(g_\xi)$. In general, any Lie group for which a volume element $d\mu$ can be found such that this is true is called *unimodular*. If the volume element is only invariant under left (or right) translations, then it is called left (or right) invariant.

B. Convolution of Functions on $SE(D)$

The Special Euclidean Group[4] [denoted here as $SE(D)$] is the $D(D+1)/2$ dimensional Lie group which describes rigid motions (rotation and translation) in D dimensional Euclidean space.

We can represent any element of the Lie group $SE(D)$ with a $(D+1)\times(D+1)$ homogeneous transform matrix of the form

$$H = \begin{pmatrix} R & \overline{b} \\ \overline{0}^T & 1 \end{pmatrix}$$

where $R \in SO(D)$ (i.e., R is an $D \times D$ special orthogonal matrix: $RR^T = 1$ and $\det(R) = +1$), $\overline{b} \in \mathbb{R}^D$, and the group operation is matrix multiplication of two homogeneous transform matrices.

Since $SE(D)$ is a Lie group, the concept of convolution reviewed in the previous section can be applied. While convolution of functions on Lie groups has been a familiar concept in the pure mathematics and theoretical physics literature for quite some time [28], [29], applications to manipulator kinematics are certainly new. Hence we dedicate this section to a more in depth discussion of the particular case when $G = SE(D)$.

The explicit computation of volume elements for $G = SE(2)$ and $SE(3)$ are given in the Appendix. $SE(2)$ and $SE(3)$ are both unimodular, as has been observed by Park and Brockett [21] and Murray, Li, and Sastry [18]. This means that the convolution of functions on the Euclidean group of either dimension can be written in either of the forms

$$\begin{aligned}(\alpha * \beta)(g_x) &= \int_G \alpha(g_\xi)\beta(g_\xi^{-1} \circ g_x)\, d\mu(g_\xi) \\ &= \int_G \alpha(g_x \circ g_\xi^{-1})\beta(g_\xi)\, d\mu(g_\xi) \end{aligned} \tag{4}$$

where the change of variables $g_\xi \to g_x \circ g_\xi^{-1}$ has been employed, and the fact that for unimodular groups

$$\begin{aligned}\int_G f(g_\xi)\, d\mu(g_\xi) &= \int_G f(g_\xi^{-1})\, d\mu(g_\xi) \\ &= \int_G f(g_x \circ g_\xi)\, d\mu(g_\xi) \\ &= \int_G f(g_\xi \circ g_x)\, d\mu(g_\xi)\end{aligned}$$

is observed.

If a function is convolved with itself, we can write

$$\begin{aligned}(\rho * \rho)(g_x) &= \int_G \rho(g_\xi)\rho(g_\xi^{-1} \circ g_x)\, d\mu(g_\xi) \\ &= \int_G \rho(g_x \circ g_\xi^{-1})\rho(g_\xi)\, d\mu(g_\xi)\end{aligned}$$

which means that the order of the product of g_x and g_ξ^{-1} does not matter in this special case even if the group is not commutative.

The fact that $d\mu(g)$ is invariant under translations gives the convolution product some other useful properties. Namely, if $\rho_i(g) \in \mathcal{L}^2(G)$ for $i = 1, 2$ are nonnegative functions for all $g \in G$ then

$$\begin{aligned}&\int_G (\rho_1 * \rho_2)(g)\, d\mu(g) \\ &\quad = \left(\int_G \rho_1(g)\, d\mu(g)\right)\left(\int_G \rho_2(g)\, d\mu(g)\right).\end{aligned}$$

This is true because

$$\begin{aligned}&\int_G (\rho_1 * \rho_2)(g_x)\, d\mu(g_x) \\ &\quad = \int_G \int_G \rho_1(g_\xi)\rho_2(g_\xi^{-1} \circ g_x)\, d\mu(g_\xi)\, d\mu(g_x) \\ &\quad = \int_G \int_G \rho_1(g_\xi)\rho_2(g_z)\, d\mu(g_\xi)\, d\mu(g_\xi \circ g_z) \\ &\quad = \left(\int_G \rho_1(g_\xi)\, d\mu(g_\xi)\right)\left(\int_G \rho_2(g_z)\, d\mu(g_z)\right).\end{aligned} \tag{5}$$

The change of variables $g_z = g_\xi^{-1} \circ g_x$ has been made, and since $d\mu(g_\xi \circ g_z) = d\mu(g_z)$, the integrals separate.

C. Geometric Interpretation of Convolution of Functions on SE(D)

Suppose there are three frames in space, F_1, F_2, and F_3, as shown in Fig. 2. The first frame can be viewed as fixed, the second frame as moving with respect to the first, and the third frame as moving with respect to the second. Let the homogeneous transform $\mathcal{H}$ describe the position and orientation of F_2 w.r.t. F_1, and H describe the position and orientation of F_3 w.r.t. F_2. Then the position and orientation of F_3 with respect

[3] Often a right translation is defined as $\beta(g_x \circ g_\xi)$ instead of $\beta(g_x \circ g_\xi^{-1})$.

[4] Also called the Euclidean motion group, or simply the Euclidean group.

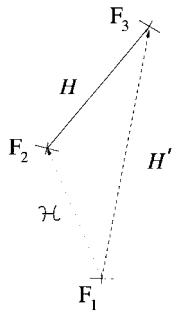

Fig. 2. Concatenation of homogeneous transformations.

to F_1 is $H' = \mathcal{H}H$. The position and orientation of F_3 with respect to F_2 can then be written as

$$H = \mathcal{H}^{-1}H'.$$

We may divide up $SE(D)$ into volume elements, or "voxels," of finite but small size. The volume of the voxel centered at $H \in SE(D)$ is denoted $\Delta\mu(H)$, and as the element size is chosen smaller and smaller it becomes closer to the differential volume element $d\mu(H)$.

The motion of F_2 relative to F_1 and the motion of F_3 relative to F_2 can both be considered elements of $SE(D)$, and no distinction is made between these motions and the transformation matrices $\mathcal{H}$ and H which represent these motions.

Assuming we move $\mathcal{H}$ and H through a finite number of different positions and orientations, let ρ_1 be a function that records how often the $\mathcal{H}$ frames appear in each voxel, divided by the voxel volume $\Delta\mu(\mathcal{H})$. Likewise, let ρ_2 be the function describing how often the H frames appear in each voxel normalized by voxel volume.

To calculate how often the H' frames appear in each voxel in $SE(D)$ for all possible values of $\mathcal{H}$ and H, we may perform the following steps.

1) Evaluate $\rho_1 = \rho_1(\mathcal{H})$ (frequency of occurrence of $\mathcal{H}$),
2) Evaluate $\rho_2 = \rho_2(H) = \rho_2(\mathcal{H}^{-1}H')$ (frequency of occurrence of $H = \mathcal{H}^{-1}H'$).
3) Weight (multiply) the left-shifted density histogram $\rho_2(\mathcal{H}^{-1}H')$ by the number of frames which are doing the shifting. This number is $\rho_1(\mathcal{H})\Delta\mu(\mathcal{H})$ for each $\mathcal{H}$.[5]
4) Sum (integrate) over all these contributions

$$(\rho_1 * \rho_2)(H') = \int_{SE(D)} \rho_1(\mathcal{H})\rho_2(\mathcal{H}^{-1}H')\, d\mu(\mathcal{H}).$$

As will be seen in subsequent sections, this approach yields an approximation to the density of H' frames which can be computed very efficiently. The number of H' frames in each voxel of $SE(D)$ can be calculated from this density as simply $(\rho_1 * \rho_2)(H')\Delta\mu(H')$.

IV. Use of Convolution for Workspace Generation

In the previous section, we showed how the concept of a convolution product of two functions can be extended to arbitrary Lie groups with an appropriate concept of integration. In this section we investigate the applications of convolution in workspace generation for the particular case of real-valued functions on $SE(D)$.

A. The General Idea

In this subsection, we show how the concept of the convolution product of functions on $SE(D)$ can be applied to the generation of workspaces.

Let us consider a manipulator that consists of two mechanisms stacked on top of one another. For example, the two mechanisms can be in-parallel platform mechanisms or serial linkages. A frame F_1 is attached to the bottom of the first mechanism and a frame F_2 to its top, which also defines the bottom of the second mechanism. A third frame, F_3, defines the position and orientation of the top of the second mechanism. This leads back to the situation shown in Fig. 2, where now $\mathcal{H}$ describes the homogeneous transformation corresponding to the lower mechanism, H the one for the upper mechanism, and H' the one for the whole manipulator.

If the manipulator is actuated discretely, then each mechanism only has a finite number of different states, which can be described by two finite sets, $\mathcal{S}_1$ and $\mathcal{S}_2$, which contain m_1 and m_2 elements, respectively. The set of all homogeneous transformations that can be attained by the distal end of the manipulator when the base is fixed results from all possible combinations of these two sets

$$\mathcal{S}' = \{H' = \mathcal{H}H\colon\ H \in \mathcal{S}_1, \mathcal{H} \in \mathcal{S}_2\}$$

and hence consists of $m_1 \cdot m_2$ elements.

For a discrete set of frames (elements of $SE(D)$) that is very large, it is useful to reduce the amount of data by approximating its information with a density function. This is done by dividing a bounded region in $SE(D)$ into small volume elements, counting how many reachable frames occupy each volume element, and dividing this number by the volume of each element. This density function describes the distribution of frames in the workspace. Fig. 1 shows the integral of such a distribution over all orientations reachable by a manipulator. The result is a function of position in the plane (gray scale corresponds to density).

If the sets $\mathcal{S}_1$ and $\mathcal{S}_2$ are approximated with density functions $\rho_1(\cdot)$ and $\rho_2(\cdot)$, respectively, then the density function resulting from the convolution of the two is a density function for the whole manipulator

$$\rho(H') = (\rho_1 * \rho_2)(H').$$

Furthermore, this reasoning can be applied to manipulators consisting of more than two mechanisms (modules) stacked on top of one another. For instance, if four modules are stacked, the density of the lower two is $\rho_1 * \rho_2$ and the density of the upper two is $\rho_3 * \rho_4$ by the reasoning given previously. Treating the lower two modules as one big module, and the

[5]Note that the product $\rho_2(\mathcal{H}^{-1}H')\rho_1(\mathcal{H})\Delta\mu(\mathcal{H})$ is then an approximation to the sum of histograms which would result by sweeping $\rho_2(H)$ by the homogeneous transforms $\mathcal{H}$ in the voxel with volume $\Delta\mu(\mathcal{H})$. In the limiting case when the volume size goes to zero, this approximation becomes exact.

upper two as one big module, the density of the collection of all four modules is $(\rho_1 * \rho_2) * (\rho_3 * \rho_4) = \rho_1 * \rho_2 * \rho_3 * \rho_4$.

If the reasonable frames of a discrete manipulator consisting of P independent modules are described by sets $S_1, \cdots, S_P$ in the way described above, then the density of the whole manipulator is derived by multiple convolution as

$$\rho(H') = (\rho_1 * \rho_2 * \cdots * \rho_P)(H').$$

B. Computational Benefit of this Approach

Suppose a manipulator consisting of P modules is considered and the number of homogeneous transformations in each set $\mathcal{S}_i$ is m_i. The explicit (brute force) calculation of all combinations of homogeneous transformations is of the same order as the number of all combinations, $\mathcal{O}(\Pi_{i=1}^{P} m_i)$. If $m_1 = \cdots = m_P = K$, then this is an $\mathcal{O}(K^P)$ calculation.

In our approach, density functions are used to describe the frame distribution for each module. The frame distribution of the whole manipulator results by performing P convolutions. These calculations also depend on the dimension of $SE(D)$ for $D = 2, 3$. While we treat D as a constant because it does not change with the number of actuator states, it is worth noting that if a compact subset of $SE(D)$ is divided into $\mathcal{N}_j$ increments in each dimension, then $Q = \Pi_{j=1}^{D(D+1)/2} \mathcal{N}_j$ voxels result. Treating Q as constant, the calculation of $\rho_i(\cdot)$ requires $\mathcal{O}(m_i)$ additions to increment the number of frames in each voxel. Since the voxels are uniform in size in the $SE(2)$ case, there is no need to explicitly divide by voxel volume (i.e., this normalization can be performed concurrently with convolution). Thus the calculation of $\rho_i(\cdot)$ is effectively $\mathcal{O}(m_i)$.

Consider the convolution of density functions of any two adjacent modules. The numerical approximation of the convolution integral evaluated at a single point in the support[6] of $(\rho_i * \rho_{i+1})(\cdot)$, becomes a sum over all voxels in the support of $\rho_i(\cdot)$. This calculation must be performed for all voxels in the support of $(\rho_i * \rho_{i+1})(\cdot)$, and so the computations required to perform one convolution are $\mathcal{O}(\text{convolution}) = \mathcal{O}(Q_i \cdot Q_i^*)$, where Q_i and Q_i^* are respectively the number of voxels in the support of $\rho_i(\cdot)$ and $(\rho_i * \rho_{i+1})(\cdot)$. If the voxel size is kept constant after convolution, then $Q_i^* > Q_i$, because the workspace of any two adjacent modules is bigger than either one individually. Treating Q_i and D as constants, the calculation of P convolutions would then be polynomial in P. The order of this polynomial would depend on D. However, if the voxel size is rescaled after convolution, so that $Q_i^* \approx Q_i$, then each convolution is $\mathcal{O}(Q_i^2) = \mathcal{O}(1)$. Hence the total order of this approach is $\mathcal{O}(\Sigma_{i=1}^{P} m_i) + P \cdot \mathcal{O}(1) = \mathcal{O}(P)$ for this data storage strategy.

In the special case of a manipulator consisting of P identical modules the number of convolutions to be performed can be further reduced by using a different strategy. In this case, the frame distribution $\rho(H')$ is calculated from P identical functions as a P-fold convolution

$$\rho = \rho_1^{(P)} = \rho_1 * \rho_1 * \cdots * \rho_1.$$

Note, that the repeated convolution of a function with itself generates the following sequence of functions:

$$\rho^{(2)} = \rho_1 * \rho_1, \quad \rho^{(4)} = \rho^{(2)} * \rho^{(2)}$$
$$\rho^{(8)} = \rho^{(4)} * \rho^{(4)}, \text{ etc.}$$

i.e., it is possible to generate $\rho^{(2)}$ by one convolution, $\rho^{(4)}$ by two convolutions, and more generally $\rho^{(2^n)}$ by n convolutions. Thus, for a manipulator with P identical modules, approximately $\mathcal{O}(\log P)$ convolutions have to be performed, which is an $\mathcal{O}(\log P)$ calculation if the number of voxels is held constant after each convolution (i.e., if we allow voxel size to grow with each convolution).

One can estimate the support of the convolution of two density functions by first making a gross over estimate and doing a very crude (low resolution) convolution. Those voxels which have zero density after convolution can be discarded, and what is left over is a closer overestimate of the support of the convolved functions. This region is smaller than the original estimate, and voxel sizes can be scaled down to get the best resolution for the allowable memory.

C. Explicit Computation of the Convolution Product of Functions on SE(2)

In the two-dimensional (2-D) case, the homogeneous transforms H' and $\mathcal{H}$ in the convolution integral can be parametrized as

$$H'(x, y, \theta) = \begin{pmatrix} \cos\theta & -\sin\theta & x \\ \sin\theta & \cos\theta & y \\ 0 & 0 & 1 \end{pmatrix}$$

and

$$\mathcal{H}(\xi, \eta, \alpha) = \begin{pmatrix} \cos\alpha & -\sin\alpha & \xi \\ \sin\alpha & \cos\alpha & \eta \\ 0 & 0 & 1 \end{pmatrix}.$$

We define a parametrized density function $\rho(x, y, \theta)$ by identifying

$$\rho(x, y, \theta) \equiv \rho(H'(x, y, \theta))$$

which leads to an explicit form of the convolution product on $SE(2)$

$$\begin{aligned}(\rho_1 * \rho_2)(x, y, \theta) &= (\rho_1 * \rho_2)(H') \\ &= \int_{SE(2)} \rho_1(\mathcal{H}) \rho_2(\mathcal{H}^{-1} H')\, d\mu(\mathcal{H}) \\ &= \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \int_{-\pi}^{\pi} \rho_1(\xi, \eta, \alpha) \rho_2((x - \xi)c\alpha + (y - \eta)s\alpha \\ &\quad - (x - \xi)s\alpha + (y - \eta)c\alpha, \theta - \alpha)\, d\xi\, d\eta\, d\alpha\end{aligned}$$

where $c\alpha = \cos\alpha$ and $s\alpha = \sin\alpha$.

[6] The support is the subset of the domain of the function on which the function has nonzero value.

In general if a subset of $SE(D)$ for $D = 2, 3$ is parametrized with $D(D+1)/2$ variables $q_1, \cdots, q_{D(D+1)/2}$, then to within a constant[7]

$$d\mu(\mathcal{H}(\overline{q})) = \det(J)\, dq_1 \cdots dq_{D(D+1)/2}$$

where J is the $D(D+1)/2 \times D(D+1)/2$ Jacobian matrix of the parametrization. In the case when $D = 2$ the determinant of the Jacobian matrix is one (see the Appendix for details).

For the following derivation we assume that ρ_1 and ρ_2 are real-valued functions on $SE(D)$ that are non-negative and bounded everywhere, and have compact support. That is, they vanish outside of a compact (closed and bounded) subset of $SE(2)$, which for simplicity is chosen of the form

$$[x_{\min}^{(j)}, x_{\max}^{(j)}] \times [y_{\min}^{(j)}, y_{\max}^{(j)}] \times [-\pi, \pi] \quad \text{for} \quad j = 1, 2.$$

The convolution product of two such functions on $SE(2)$ can be expressed in the form

$$\begin{aligned}(f_1 * f_2)(x_1, x_2, \theta) &= \int_{x_{\min}^{(1)}}^{x_{\max}^{(1)}} \int_{y_{\min}^{(1)}}^{y_{\max}^{(1)}} \int_{-\pi}^{\pi} \rho_1(\xi, \eta, \alpha) \\ &\quad \cdot \rho_2((x-\xi)\cos\alpha + (y-\eta)\sin\alpha, \\ &\quad -(x-\xi)\cos\alpha + (y-\eta)\sin\alpha, (\theta-\alpha) \bmod 2\pi) \\ &\quad \cdot d\xi\, d\eta\, d\alpha\end{aligned}$$

where $\bmod 2\pi$ is used here to mean that the difference $\theta - \alpha$ is taken in the range $-\pi$ to π.[8]

V. Workspace Generation for Planar Manipulators

In Section III, we showed how the concept of the convolution product of functions on Lie groups is defined. In Section IV, we showed how convolution of functions on $SE(D)$ can be applied to workspace generation. This section describes the details of a numerical implementation which is based on a representation of density as a piecewise constant histogram.

We start this section with a summary of the procedure for generating the workspace of a discretely actuated planar manipulator.

1) The manipulator is divided into P kinematically independent modules. The modules are numbered from 1 to P, starting at the base with module 1 and increasing up to the most distal module, module P. For each module one frame is attached to the base of the module and a second one to the top, where the next module is attached. Modules can have a parallel kinematic structure internally, but the modules are all cascaded in a serial way.
2) For each module, $(p = 1, \cdots, P)$ the discrete set $\mathcal{S}_p = \{H_{i_p}\}$ of all frames the top of module p can attain relative to the bottom is determined. That is, the position and orientation of the upper frame with respect to the lower frame is described by homogeneous transformations, $H_{i_p} \in \mathcal{S}_p \subset SE(2)$, while the module undergoes all possible discrete configurations.
3) A compact subset $\mathcal{C}_p \subset SE(2)$ is chosen that contains all of the discrete sets $\mathcal{S}_p$. The subset $\mathcal{C}_p$ is discretized and a piecewise constant density function/histogram ρ_p is calculated for each $\mathcal{S}_p$ to represent this information.
4) Finally, the discrete density functions are convolved in the order

$$\rho_W = \rho_1 * \rho_2 * \cdots * \rho_P$$

to yield the density function of the workspace.

The implementation of Step 1 and 2 of the workspace generation procedure depends on the architecture of the manipulator. Generating these sets is a simple task if the manipulator can be separated into a sufficiently large number of kinematically independent modules of simple structure. We assume that the discrete sets $\mathcal{S}_p$ are calculated efficiently either numerically or in closed form for all modules of the manipulator.

For Steps 3 and 4 we have to define discretized density functions used to represent each set $\mathcal{S}_p$. The discretization of the parameter space is described in the following subsection for the case $SE(2)$, and the subsection after that describes the resulting discrete form of the convolution.

A. Discretization of Parameter Space

For each set $\mathcal{S}_p$ arising in Step 3 of the procedure, the support of ρ_P in terms of the parameters (x, y, θ) is of the form $[x_{\min}^{(p)}, x_{\max}^{(p)}] \times [y_{\min}^{(p)}, y_{\max}^{(p)}] \times [-\pi, \pi]$. This set of parameters is divided into elements/voxels of equal size. Note, that in the following discussion the superscript "$^{(p)}$" is dropped, unless we refer to a particular set $\mathcal{S}_p$. We choose the resolution in the x and y-directions to be identical, i.e., $\Delta x = \Delta y$, and explain below how to choose the resolution in the angular direction such that the resulting errors from inaccuracy in position and rotation are of the same order.

We denote by N_1, N_2, and M the number of discretizations in the x-, y-, and θ-directions, respectively, i.e., $\Delta x = (x_{\max} - x_{\min})/N_1, \Delta y = (y_{\max} - y_{\min})/N_2, \Delta\theta = (2\pi - 0)/M$, and choose N_1 and N_2 such that $\Delta x = \Delta y$.[9] Each voxel is a volume element of the form $[x_{\min} + i\Delta x, x_{\min} + (i+1)\Delta x] \times [y_{\min} + j\Delta y, y_{\min} + (j+1)\Delta y] \times [-\pi + k\Delta\theta, -\pi + (k+1)\Delta\theta]$ with center coordinates $(x_i, y_j, \theta_k) = (x_{\min} + (i + 0.5)\Delta x, y_{\min} + (j + 0.5)\Delta y, -\pi + (k + 0.5)\Delta\theta)$.

To characterize the error resulting from discretization we consider a homogeneous transform $H \in SE(2)$ corresponding to some exact parameters: $H = H(x, y, \theta)$. H is then compared to the homogeneous transformation $\hat{H}$ corresponding to the rounded coordinates: $\hat{H} = H(\hat{x}, \hat{y}, \hat{\theta})$. By definition (x, y, θ) differs from $(\hat{x}, \hat{y}, \hat{\theta})$ at most by $(\Delta x/2, \Delta y/2, \Delta\theta/2)$.

If we apply H and $\hat{H}$ to any vector $v \in \mathbb{R}^2$ and use (R, b) and $(\hat{R}, \hat{b})$ to denote rotation and translation of H and $\hat{H}$

[7] For compact groups the constant is set so that $\int_G d\mu(g) = 1$, but $SE(D)$ is not compact and so there is no unique way to scale the volume element.

[8] This is different than the standard definition which would put the result in $[0, 2\pi)$.

[9] To get exact equality it is usually necessary to slightly change one of the workspace boundaries, e.g., to slightly increase $y_{\max}$.

respectively, then the error $||Hv - \hat{H}v||$ is bounded as

$$\begin{aligned}||Hv - \hat{H}v|| &= ||(Rv + b) - (\hat{R}v - \hat{b})|| \\ &= ||(Rv - \hat{R}v) + (b - \hat{b})|| \\ &\leq \underbrace{||Rv - \hat{R}v||}_{\text{rot. part}} + \underbrace{||b - \hat{b}||}_{\text{transl. part}} \\ &\leq \frac{\Delta\theta}{2}||v|| + \left(\left(\frac{\Delta x}{2}\right)^2 + \left(\frac{\Delta y}{2}\right)^2\right)^{1/2}.\end{aligned}$$

If V is a set of vectors, then the maximal difference in displacement between transformed versions of a vector $v \in V$ after transformation by H and $\hat{H}$ is bounded by

$$\begin{aligned}&\max_{v\in V} ||Hv - \hat{H}v|| \\ &\quad= \frac{\Delta\theta}{2}\max_{v\in V}||v|| + \left(\left(\frac{\Delta x}{2}\right)^2 + \left(\frac{\Delta y}{2}\right)^2\right)^{1/2} \\ &\quad= \frac{\Delta\theta}{2}\max_{v\in V}||v|| + \frac{\Delta x}{\sqrt{2}}. \qquad (6)\end{aligned}$$

The last equality holds because $\Delta x = \Delta y$.

In the case of convolution of two workspace densities we can use (6) to balance the error between the angular and translational part. We denote continuous regions containing the sets $\mathcal{S}_1$ and $\mathcal{S}_2$ as $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively. To discretize $\mathcal{C}_1$ we first select a maximal acceptable error e (which results from a trade-off between memory and accuracy). The resolution parameters $\Delta x^{(1)}, \Delta y^{(1)}, \Delta\theta^{(1)}$ of $\mathcal{C}_1$ are then determined such that the two parts of the error are of the same order and add up to e, i.e.

$$\frac{\Delta\theta^{(1)}}{2}\max_{v\in T(\mathcal{C}_2)}||v|| \stackrel{!}{=} \frac{\Delta x^{(1)}}{\sqrt{2}} \stackrel{!}{=} \frac{e}{2}$$

where $T(\mathcal{C}) \subset \mathbb{R}^2$ is the union of all projections of constant theta slices of $\mathcal{C}$ onto the plane.

This results in the choice

$$\Delta x^{(1)} = \Delta y^{(1)} = \frac{e}{\sqrt{2}}, \qquad \Delta\theta^{(1)} = \frac{e}{\max_{v\in T(\mathcal{C}_2)}||v||}. \qquad (7)$$

The step sizes for the discretization of $\mathcal{C}_2$ are chosen analogously.

B. Numerical Convolution of Histograms on SE(2)

Our goal is to store density functions in the form of piecewise constant histograms, i.e. we only want to store average values for each voxel. This section presents convolution in a form applicable to histograms on $SE(2)$. As a first step the integral from the previous section

$$\begin{aligned}f_3(x,y,\theta) &= (f_1 * f_2)(x,y,\theta) \\ &= \int_{x_{\min}^{(1)}}^{x_{\max}^{(1)}}\int_{y_{\min}^{(1)}}^{y_{\max}^{(1)}}\int_{-\pi}^{\pi} f_1(\xi,\eta,\alpha) \\ &\quad\cdot f_2((x-\xi)\cos\alpha + (y-\eta)\sin\alpha, \\ &\quad -(x-\xi)\sin\alpha + (y-\eta)\cos\alpha, \\ &\quad (\theta-\alpha) \bmod 2\pi)\, d\xi\, d\eta\, d\alpha\end{aligned}$$

is approximated by a Riemann–Stieltjes sum

$$\begin{aligned}&(f_1 * f_2)(x,y,\theta) \\ &\quad\approx \sum_{l=0}^{N_1}\sum_{m=0}^{N_2}\sum_{n=0}^{M} f_1(\xi_l,\eta_m,\alpha_n)\cdot f_2((x-\xi_l)\cos\alpha_n \\ &\quad + (y-\eta_m)\sin\alpha_n, -(x-\xi_l)\sin\alpha_n + (y-\eta_m) \\ &\quad \cdot\cos\alpha_n, (\theta-\alpha_n) \bmod 2\pi)\,\Delta\xi\,\Delta\eta\,\Delta\alpha.\end{aligned}$$

Although the right hand side of the equation is approximated by a discrete sum, the function f_2 is required to exist for any arguments because its arguments generally do not coincide with points on the grid of any discretization. We therefore approximate the functions f_2 for any real-valued arguments by interpolation using function values at neighboring discrete points. Since this problem is frequently encountered in many applications, there exist many different strategies. The simplest strategy is that the function $f_2(x,y,\theta)$ is approximated by the value of $f_2(x_i,y_j,\theta_k)$ (i.e., at the closest point on the grid). Because this can lead to large round-off errors, we instead use linear interpolation. For each coordinate (x,y,θ) we find the indices i,j,k in the grid such that $x_i \leq x \leq x_{i+1}, y_j \leq y \leq y_{j+1}$, etc., and define the ratios

$$t = \frac{x - x_i}{\Delta x}, \quad u = \frac{y - y_j}{\Delta y}, \qquad v = \frac{\theta - \theta_k}{\Delta\theta}.$$

The value $f_2(x,y,\theta)$ is then interpolated from the values at eight discrete points (8-point interpolation)

$$\begin{aligned}\hat{f}_2(x,y,\theta) &= (1-t)(1-u)(1-v)f_2(x_i,y_j,\theta_k) \\ &\quad + (t)(1-u)(1-v)f_2(x_{i+1},y_j,\theta_k) \\ &\quad + (1-t)(u)(1-v)f_2(x_i,y_{j+1},\theta_k) + \cdots \\ &\quad + (t)(u)(v)f_2(x_{i+1},y_{j+1},\theta_{k+1}).\end{aligned}$$

C. Convolution of Functions on SE(3)

The efficient implementation of convolution of functions on $SE(3)$ is a topic of current research [6], [15]. In this subsection we briefly describe why this case is more difficult than $SE(2)$ and provide an example which illustrates how these difficulties can be avoided in certain cases.

The convolution product for functions on $SE(3)$ can be derived explicitly in a very similar way as for functions on $SE(2)$. We may parametrize the homogeneous transforms H' and $\mathcal{H}$ as

$$H' = \begin{pmatrix} R & \overline{x} \\ \overline{0}^T & 1\end{pmatrix} \quad \mathcal{H} = \begin{pmatrix} \mathcal{R} & \overline{\xi} \\ \overline{0}^T & 1\end{pmatrix}$$

where $R = R(\phi_1,\phi_2,\phi_3)$ and $\mathcal{R} = \mathcal{R}(\psi_1,\psi_2,\psi_3)$ are rotation matrices parametrized by Euler angles, and $\overline{x}, \overline{\xi} \in \mathbb{R}^3$.

For any function $\rho(H')$ we may use the slightly different notation $\rho(R,\overline{x})$. Therefore, since

$$\begin{aligned}\mathcal{H}^{-1}H' &= \begin{pmatrix} \mathcal{R}^T & -\mathcal{R}^T\overline{\xi} \\ \overline{0}^T & 1\end{pmatrix}\begin{pmatrix} R & \overline{x} \\ \overline{0}^T & 1\end{pmatrix} \\ &= \begin{pmatrix} \mathcal{R}^T R & \mathcal{R}^T(\overline{x} - \overline{\xi}) \\ \overline{0}^T & 1\end{pmatrix}\end{aligned}$$

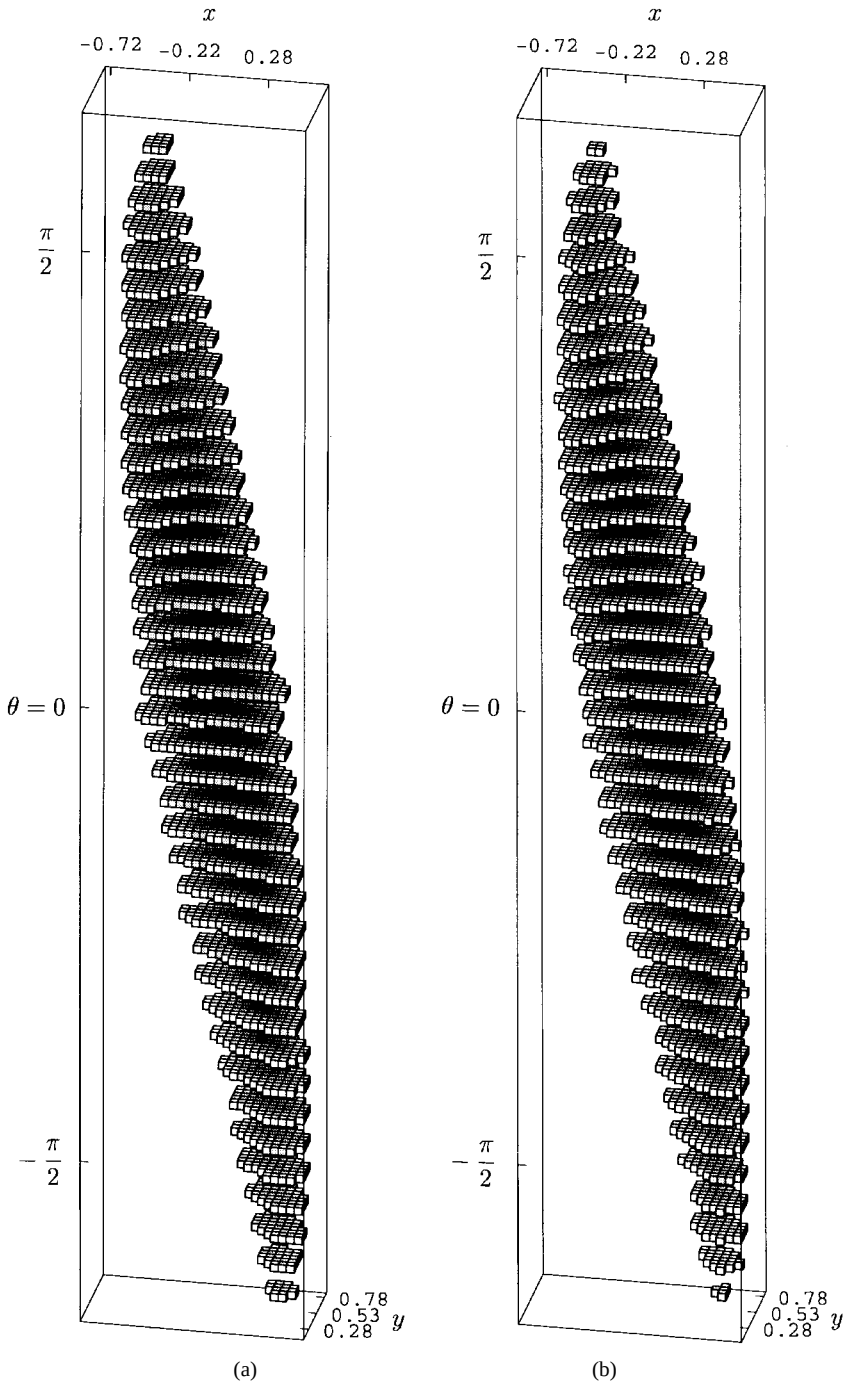

Fig. 3. Workspace density for a four-module manipulator (a) calculated by brute force; (b) using convolution. Scale is $\Delta x = \Delta y = 0.05$ and $\Delta\theta = \pi/30$.

the convolution integral may be written as

$$(\rho_1 * \rho_2)(R, \overline{x}) = \int_{SO(3)} \int_{\mathbb{R}^3} \rho_1(\mathcal{R}, \overline{\xi}) \rho_2(\mathcal{R}^T R, \mathcal{R}^T(\overline{x} - \overline{\xi}))\, d\overline{\xi}\, d\mathcal{R}. \tag{8}$$

In the above equation the integration over $SE(3)$ has been rewritten as integration over position and orientation separately, and the volume element is rewritten by observing that it is the product of the volume element for $SO(3)$ and $\mathbb{R}^3$. That is, to within a constant factor $d\mu(\mathcal{H}) = d\overline{\xi}\, d\mathcal{R}$ where $d\overline{\xi} = d\xi_1\, d\xi_2\, d\xi_3$ and $d\mathcal{R}$ is calculated in the Appendix.

There are two major difficulties that arise in the three dimensional case.

First of all, the number of parameters/dimensions rises from three to six, when moving from $SE(2)$ to $SE(3)$. This means that if the density functions are stored as arrays, the amount of memory needed is **squared**, as compared to the case of $SE(2)$ for the same discretization. This drastically reduces the allowable resolution.

Secondly, when parametrizing $SE(3)$ the choice of the discretization becomes tricky because as a manifold, $SE(3) \cong \mathbb{R}^3 \times SO(3)$, which means that each finite but small volume element in $SE(3)$ is the Cartesian product of one in $\mathbb{R}^3$

and $SO(3)$. Difficulties arise because unlike $SO(2)$, it is not possible to finely and uniformly discretize $SO(3)$. In fact, the collection of center points of any such discretization would correspond to the finite subgroups of $SO(3)$, and it is well known that these subgroups contain relatively few elements [16].

The second problem above may be addressed with more sophisticated interpolation schemes which interpolate the $SO(3)$ part of the convolution as well as the $\mathbb{R}^3$ part. If we are interested only in determining the shape of the workspace, reduction of the resolution may not be a terrible problem either. However, if we are interested in high resolution in the three dimensional case, the workspaces of only very special manipulator morphologies can be generated using convolution at the current time due to limitations on computer memory. We therefore examine one of the scenarios in which convolution can be used at the current time.

Consider a spatial discretely actuated manipulator which consists of two modules, a base (proximal) module which only translates the frame attached to its top relative to its base, and a distal module which is completely general. In this case, the density function of the base has a very special form. Since all the density is concentrated in the translation subgroup of $SE(3)$, we may write

$$\rho_1(\mathcal{R},\overline{\xi}) = \hat{\rho}_1(\overline{\xi})\delta(\mathcal{R})$$

where $\delta(R)$ is the Dirac delta function on $SO(3)$. That is, the function $\delta(R)$ is a singular distribution centered at the identity element which has the properties

$$\int_{SO(3)} \delta(\mathcal{R})\, d\mathcal{R} = 1$$

$$\int_{SO(3)} f(\mathcal{R})\delta(\mathcal{R}^T R)\, d\mathcal{R} = f(R)$$

$$\delta(\mathcal{R}^T R) = \delta(R^T \mathcal{R}).$$

Using these properties, we may rewrite (8) in the following way for this special case:

$$(\rho_1 * \rho_2)(R,\overline{x}) = \int_{\mathbb{R}^3} \hat{\rho}_1(\overline{\xi})\rho_2(R,\overline{x}-\overline{\xi})\, d\overline{\xi}. \qquad (9)$$

While this reduces a six-dimensional integration to a three-dimensional one, it does not solve the problem of memory allocation. The only way to reduce the memory requirement is to compress the information from the six dimensions of $SE(3)$ to a lower dimensional space, i.e., if we are satisfied in knowing positional density instead of that of both position and orientation. This is achieved by the integration

$$\hat{\rho}_3(\overline{x}) = \int_{SO(3)} (\rho_1 * \rho_2)(R,\overline{x})\, dR$$

$$= \int_{\mathbb{R}^3} \hat{\rho}_1(\overline{\xi}) \int_{SO(3)} \rho_2(R,\overline{x}-\overline{\xi})\, d\overline{\xi}\, dR. \qquad (10)$$

Now if we are only interested in positional density, it would be a considerable waste of memory to store $\rho_2(R,\overline{x})$ in six dimensions, only to reduce it to three by integration. Therefore, we may define from the beginning the function

$$\hat{\rho}_2(\overline{x}) = \int_{SO(3)} \rho_2(R,\overline{x})\, dR$$

which may be calculated directly as in [7]. Then (9) is written as

$$\hat{\rho}_3(\overline{x}) = \int_{\mathbb{R}^3} \hat{\rho}_1(\overline{\xi})\hat{\rho}_2(\overline{x}-\overline{\xi})\, d\overline{\xi}$$

which is the standard convolution of functions in $\mathbb{R}^3$. In this way, only three dimensional arrays need to be stored at any instance.

VI. Numerical Results for Planar Workspace Generation

In this section we present numerical results for the generation of workspaces using the methods presented earlier in this paper. The algorithm is implemented on a SUN SPARCstation 5, 110 MHz, in the C programming language. Figures were made using Mathematica version 3.0. The algorithm is applied to a version of the discretely actuated manipulator shown in Fig. 1. The manipulator in Fig. 1 consists of ten modules composed of three legs each, where each leg has two bits (four states). Hence each module has $4^3 = 64$ discrete states. In our example we consider a manipulator consisting of only eight identical modules of this kind, resulting in a manipulator with $(64)^8 \approx 2.8 \cdot 10^{14}$ states. Unless specified otherwise, the width of each platform is chosen as $w = 0.2$ compared to the minimal and maximal actuator lengths of $q_1 = 0.15$ and $q_2 = 0.22$.

For this manipulator we calculate the workspace density corresponding to only the first two modules by brute force ($64^2 = 4096$ states), which we will refer to as W2 in the following. Since all modules are identical, convolution of this density with itself leads to the density of the four-module workspace, W4. Convolving this workspace again with itself leads to the workspace density of the whole manipulator, W8.

For the workspace of four modules it is possible to calculate the results using brute force [$(64)^4 \approx 1.7 \cdot 10^7$ states]. In the following we first compare the results of this approach to the results obtained from convolution and quantify the error. Afterwards we show results for the workspace of the eight-module manipulator, which cannot simply be calculated by brute force ($2.8 \cdot 10^{14}$ states).

Fig. 3(a) is generated by calculating the histogram directly (brute force) and Fig. 3(b) is generated by convolution of the density function corresponding to two modules with itself. In the brute force calculation we use linear (eight-point) interpolation when incrementing voxels, because this makes the raw data less sensitive to small shifts in the grid. Eight-point interpolation is also used when evaluating the discrete density functions for the discrete convolution.

In each figure, the z-axis corresponds to the angle θ (orientation of end-effector), and the point density per voxel is represented by a gray scale (black representing very high density). The base of the manipulator lies at the origin of the coordinate system. To enhance differences in low density

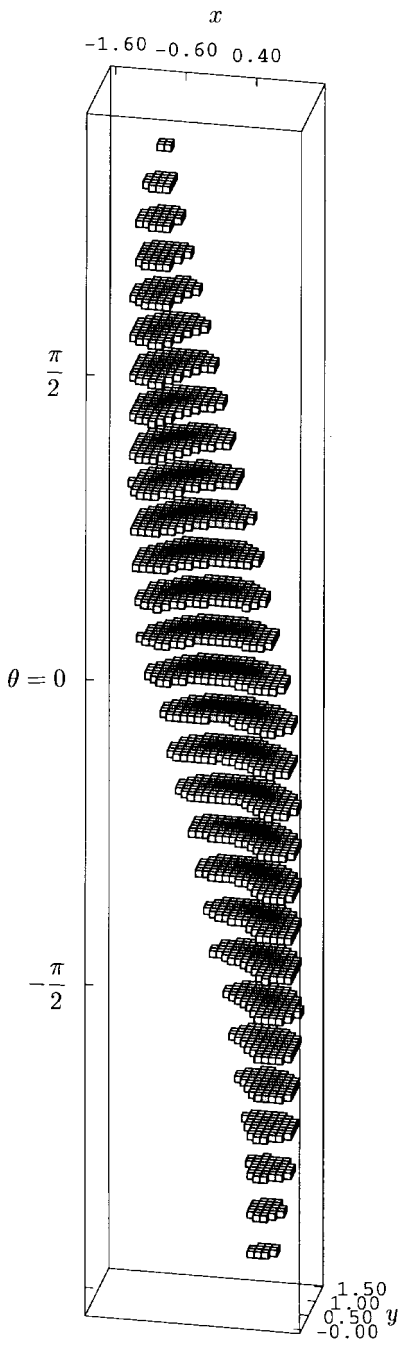

Fig. 4. Workspace density for an eight-module manipulator using convolution. Scale is $\Delta x = \Delta y = 0.1$ and $\Delta\theta = \pi/15$.

areas, we chose a nonlinear gray scale: Each density value is normalized to a value between 0 and 1 (divided by the largest density value in the drawing), and the fourth root of this value is displayed as gray value for W4 (the eighth root for W8).

Convolving the density array in Fig. 3(a) with itself results in W8. The result is shown in Fig. 4.

Fig. 5(a) and (b) show the workspace of the same manipulator as in Fig. 4, if the maximal actuator length is decreased to $q_2 = 0.2$ or increased to $q_2 = 0.25$, respectively.

Error measures: To quantify the error resulting from convolution for W4 (as compared to direct calculation) three different error measures are used. The first two measures compare the density resulting from the brute force approach, ρ, with the density from numerical convolution, $\hat{\rho}$

$$E_1 = \frac{\sum_{i,j,k} |\rho(x_i, y_j, \theta_k) - \hat{\rho}(x_i, y_j, \theta_k)|}{\sum_{i,j,k} |\rho(x_i, y_j, \theta_k)|}$$

$$E_2 = \frac{\sum_{i,j,k} |\rho(x_i, y_j, \theta_k) - \hat{\rho}(x_i, y_j, \theta_k)|^2}{\sum_{i,j,k} |\rho(x_i, y_j, \theta_k)|^2}.$$

The third measure is a function of the shapes of the workspaces (by shape we mean the set of all voxels with nonzero density), by counting the number of voxels which belong to one of the workspaces, but not to the other:

$$E_3 = \frac{\#\text{for which } ((\rho > 0) \& (\hat{\rho} = 0)) \text{ or } ((\rho = 0) \& (\hat{\rho} > 0))}{\#\text{for which } (\rho > 0).}$$

Tables I and II show error, memory and time, for the approximation of the manipulator of four modules described above, if different discretizations are chosen for the representation of the workspace. In particular, the time listed is the net time to calculate workspace W4 by one numerical convolution of W2 with itself. The memory listed is the amount of memory needed to represent either W2 or W4 (whichever has more voxels). Since voxels are consolidated after convolution, it is possible for W4 to require fewer voxels than W2.

In all simulations we chose the discretization as follows:

1) choose Δx_2 for workspace W2 and Δx_4 for W4;
2) calculate the angular discretization, $\Delta\theta_2$, of W2 according to (7);
3) define the scale factor f as $f = \Delta x_4 / \Delta x_2$, and determine $\Delta\theta_4$ such that f is also the factor for the angular discretization, i.e. $\Delta\theta_4 = f \cdot \Delta\theta_2$. As a practical matter, integer scale factors are the easiest to work with. However it is possible to consolidate voxels using more general scale factors if interpolation is used.

Results and Analysis:

1) For the case considered, W4 can be calculated from W2 with one convolution in less than 12 min with an error smaller than 10%. W8 is calculated from W4 with one convolution within another 9 min. (No error is reported, since comparison with brute force results is impossible for W8).
2) The best results in this particular example were obtained using a factor of $f \approx 2$, i.e. if the resolution is twice as fine before convolution than after (see Table I). In general, an appropriate scale factor can be estimated even if there is no brute force data against which to compare the results of convolution. This may be achieved by examining the relative error of sequences

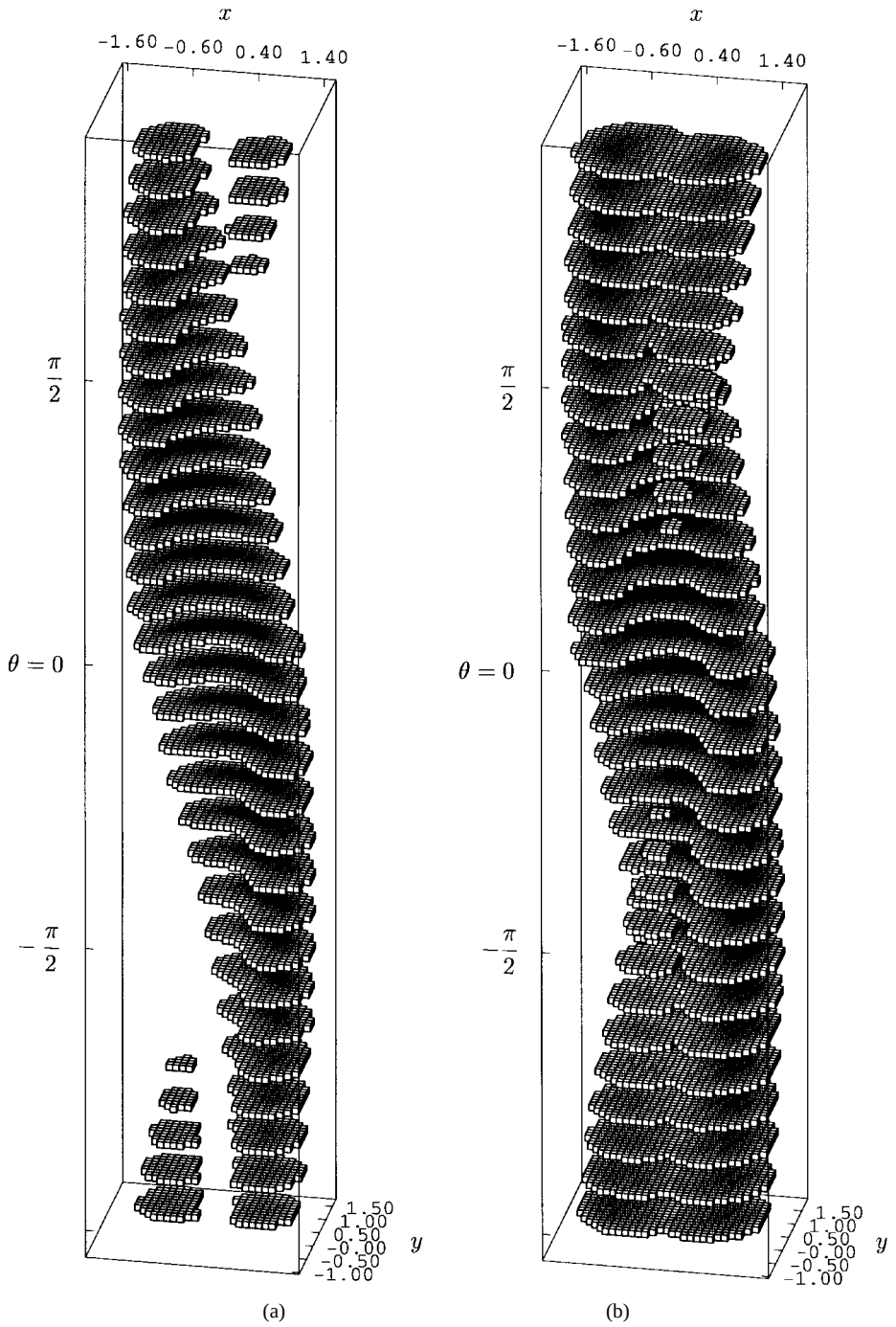

Fig. 5. Workspace density for an eight-module manipulator with kinematic parameters (a) $q_2 = 0.2$, (b) $q_2 = 0.25$. Scale is $\Delta x = \Delta y = 0.1$ and $\Delta\theta = \pi/15$.

TABLE I
RESULTS FOR FIXED DISCRETIZATION $(\Delta x_4, \Delta\theta_4)$ AND VARYING FACTOR f

factor f	$(\Delta x_2 \ , \ \Delta\theta_2)$	$(\Delta x_4 \ , \ \Delta\theta_4)$	E_1	E_2	E_3	max. memory	Time
1	(0.0500, 0.101)	(0.050, 0.101)	18.95%	17.08%	96.42%	48 KB	2.4 min
2	(0.0250, 0.051)	(0.050, 0.101)	8.27%	9.64%	9.42%	442 KB	11.5 min
4	(0.0125, 0.025)	(0.050, 0.101)	21.82%	27.05%	30.18%	3546 KB	70.8 min

of density functions generated using convolution, and picking the scale factor for which the relative error is smallest.

3) For a constant factor $f = 2$ the error decreases slowly with higher discretization (see Table II).

4) The result of the discrete convolution is closer to a brute force calculation if eight-point interpolation is used to generate the brute force results, as compared to no interpolation. Table III lists the error between brute force (BF) and discrete (numerical) convolution (DC) if

TABLE II
RESULTS FOR FIXED FACTOR $f = 2$ AND VARYING DISCRETIZATION ($\Delta x_4, \Delta\theta_4$)

factor f	$(\Delta x_2 \ , \ \Delta\theta_2)$	$(\Delta x_4 \ , \ \Delta\theta_4)$	E_1	E_2	E_3	max. memory	Time
2	(0.0500, 0.101)	(0.1000, 0.203)	13.77%	15.37%	13.65%	54 KB	23 sec
2	(0.0375, 0.077)	(0.0750, 0.153)	7.48%	7.91%	13.58%	131 KB	91 sec
2	(0.0250, 0.051)	(0.0500, 0.101)	8.27%	9.64%	9.42%	442 KB	11.5 min
2	(0.0188, 0.038)	(0.0375, 0.076)	7.46%	9.05%	7.68%	1056 KB	47.1 min
2	(0.0125, 0.025)	(0.0250, 0.050)	5.20%	6.18%	6.47%	3546 KB	7hrs 11min

TABLE III
RESULTS FOR DIFFERENT TYPES OF INTERPOLATION AND DIFFERENT NUMBER OF STATES PER ACTUATOR

Four states per actuator:

$(\Delta x_2 \ , \ \Delta\theta_2)$	$(\Delta x_4 \ , \ \Delta\theta_4)$	brute force	disc. conv.	E_1	E_2	E_3	T_{BF}	T_{DC}
(0.0200, 0.041)	(0.040, 0.082)	1-point	2-point	10.56%	13.26%	13.58%	4.8 min	12.7 min
(0.0200, 0.041)	(0.040, 0.082)	1-point	8-point	10.90%	14.56%	9.83%	4.8 min	20.8 min
(0.0200, 0.041)	(0.040, 0.082)	8-point	8-point	7.72%	9.58%	8.27%	12.1 min	33.9 min
(0.0250, 0.051)	(0.050, 0.101)	1-point	2-point	11.91%	15.24%	16.74%	4.8 min	4.3 min
(0.0250, 0.051)	(0.050, 0.101)	1-point	8-point	9.78%	12.58%	9.60%	4.8 min	7.2 min
(0.0250, 0.051)	(0.050, 0.101)	8-point	8-point	8.27%	9.64%	9.42%	16.8 min	11.5 min

Two states per actuator:

$(\Delta x_2 \ , \ \Delta\theta_2)$	$(\Delta x_4 \ , \ \Delta\theta_4)$	brute force	disc. conv.	E_1	E_2	E_3	T_{BF}	T_{DC}
(0.0250, 0.051)	(0.050, 0.101)	1-point	2-point	151.22%	235.52%	75.43%	0.1 sec	3.4 min
(0.0250, 0.051)	(0.050, 0.101)	1-point	8-point	121.50%	158.80%	62.51%	0.3 sec	12.5 min
(0.0250, 0.051)	(0.050, 0.101)	8-point	8-point	37.70%	21.64%	21.64%	0.7 sec	20.3 min

different numbers of points are used in the interpolation, together with the corresponding computation time for either method. (One-point means no interpolation at all, two-point means interpolation only in the angle, eight-point means interpolation in all three coordinate axes of the grid.)

We expected the error to reduce more or less monotonically with finer discretization. As can be seen in Table II this is true for the shape error, E_3. For error E_1 and E_2 the general tendency is to decrease for higher resolution, but this does not happen strictly monotonically. One likely reason for this is sensitivity of density to shifting of the grid by a tiny amount. This effect only appears if the density is distributed very unevenly and differs considerably in neighboring voxels. Hence we expect this effect also to disappear for manipulators of higher resolution, i.e. if the actuators have a higher resolution or a larger number of modules are considered as a single unit. This expectation is supported by the data in Table III which indicates a dramatic convergence of results obtained by brute force and convolution as the number of actuator states in each leg is doubled.

It is worth noting that for manipulators composed of P modules which each have a very large number of states, the $\mathcal{O}(P)$ or $\mathcal{O}(\log P)$ calculations required for P convolutions can be a significant savings over $\mathcal{O}(K^p)$. However, in the example presented here, the 64 frames reachable by each module are relatively sparse. This means that the actual time required to perform the $\log_2(8) = 3$ convolutions would far exceed the time required to generate the density function for half of the manipulator (which requires $(64)^{8/2}$ kinematic calculations) and perform one convolution. Thus in this case, $\mathcal{O}(K^{P/2})$ is better than $\mathcal{O}(\log P)$. In either case convolution plays a critical role in avoiding the $\mathcal{O}(K^P)$ calculations which cannot be performed in a reasonable amount of time.

Finally, we note that the purpose of this implementation is to show that the concept of numerical convolution on $SE(D)$ works and provides usable results. Since this approach is new, and hence not optimized, it is likely that more accurate and more efficient implementations will be formulated in the future.

VII. CONCLUSION

In this work, it was shown that the general concept of convolution is applicable to the approximation of workspaces and workspace densities of hybrid serial-parallel manipulators. The mathematical properties of the convolution product of scalar functions on the Euclidean motion group were reviewed and a numerical scheme, based on convolution, was generated.

Further research will address the implementation of convolution for continuous functions and the generalization of the Fourier transform to functions on the Euclidean motion group as a means of performing convolution quickly. The concept of convolution is also promising as a tool for formulating the workspace synthesis problem for discretely actuated manipulators.

APPENDIX: INTEGRATION

Let

$$g(\overline{q}) = \begin{bmatrix} R & \overline{x} \\ \overline{0}^T & 1 \end{bmatrix} \in SE(D)$$

denote the displacement (i.e., homogeneous $(D+1)\times(D+1)$ matrix) describing an arbitrary rigid motion in D dimensional Euclidean space, where $\overline{q} \in \mathbb{R}^{D(D+1)/2}$ parametrizes $SE(D)$. In the discussion that follows $D = 2,3$ since these are the only cases of interest in this paper.

In general, for an object whose location in space is given by a displacement $g(t)$, its *spatial velocity* is computed as

$$\dot{g}g^{-1}$$

where the "˙" denotes differentiation with respect to time. Note that the matrix $\dot{g}g^{-1}$ takes the form:

$$\begin{bmatrix} \dot{R}R^T & \dot{\overline{x}} - \dot{R}R^T\overline{x} \\ \overline{0}^T & 1 \end{bmatrix} = \begin{bmatrix} \hat{\omega} & \overline{v} \\ \overline{0}^T & 0 \end{bmatrix}$$

where $\hat{\omega}$ is a $D \times D$ skew symmetric matrix. We can convert this to $D(D+1)/2 \times 1$ "twist" coordinates via the "$\vee$" operator:

$$\begin{bmatrix} \overline{v} \\ \overline{\omega} \end{bmatrix} = \begin{bmatrix} \hat{\omega} & \overline{v} \\ \overline{0}^T & 0 \end{bmatrix}^{\vee}.$$

In the case when $D = 3, \overline{\omega}$ is defined such that $\hat{\omega}\overline{c} = \overline{\omega} \times \overline{c}$ for any $\overline{c} \in \mathbb{R}^3$. In the case when $D = 2, \hat{\omega}$ has the form $\begin{bmatrix} 0 & -\omega \\ \omega & 0 \end{bmatrix}$ and $\overline{\omega}$ becomes the scalar ω.

The twist vector is written as a Jacobian matrix multiplying a vector of derivatives of the parameters, $\overline{q}$, as follows:

$$\begin{bmatrix} \overline{v} \\ \overline{\omega} \end{bmatrix} = J\dot{\overline{q}}$$

where

$$J = J(g(\overline{q})) = \left[\left(\frac{\partial g}{\partial q_1} g^{-1} \right)^{\vee} \cdots \left(\frac{\partial g}{\partial q_{D(D+1)/2}} g^{-1} \right)^{\vee} \right].$$

The volume element for $SE(D)$ may be expressed using this Jacobian as:

$$d\mu(g(\overline{q})) = |\det(J)|\, dq_1 \cdots dq_{D(D+1)/2}$$

to within a free constant, which can be chosen arbitrarily provided that

$$\int_{SE(D)} \rho(g)\, d\mu(g) = M$$

where M is the "mass" corresponding to the positive real-valued density function $\rho(H)$. For instance, if the density function represents the points reachable by a discrete manipulator with P joints each with K states, $M = K^P$. This does not fix the constant since $\rho' = \rho/c$ and $d\mu' = c\, d\mu$ yield the same resulting mass, but once chosen its use must be consistent.

The fact that this volume form (which is an example of the more general concept of a Haar measure [19]) is invariant to right and left translations, i.e,

$$d\mu(g(\overline{q})) = d\mu(h \circ g(\overline{q})) = d\mu(g(\overline{q}) \circ h)$$

is well known [21], [18]. The fact that unimodular groups admit only one Haar measure (upto a one-parameter scaling) is also well known.

In particular, for the case of $D = 2$, $\overline{9}=[X_1, X_2, \theta]^T$, $\det(J)= 1$ and

$$d\mu(g(\overline{q})) = dx_1\, dx_2\, d\theta.$$

It is trivial to show that this is left and right invariant by direct calculation.

For $D = 3$, the rotation matrix is parametrized using Z-X-Z Euler-Angles as follows.

$$g(x_1, x_2, x_3, \phi, \theta, \psi) = \begin{pmatrix} R(\phi, \theta, \psi) & \overline{x} \\ \overline{0}^T & 1 \end{pmatrix}$$

where $\overline{x} = [x_1, x_2, x_3]^T$. One then calculates that

$$\det(J) = \sin\theta,$$

and so

$$d\mu(g(\overline{q})) = \sin\theta\, d\phi\, d\theta\, d\psi\, dx_1\, dx_2\, dx_3,$$

which is the product of the volume elements for $\mathbb{R}^3$ ($d\overline{x} = dx_1\, dx_2\, dx_3$, and and for $SO(3)$ ($dR = \sin\theta\, d\phi\, d\theta\, d\psi$).

In higher dimensions, the volume element for $SE(D)$ will also be the product of the one for $SO(D)$ and $\mathbb{R}^D$, but this is not of use to us in the current presentation. Readers interested in different perspectives on the volume element for $SE(D)$ should consult [2], [18], [21], [29].

The invariance of this volume element is seen as follows. Right invariance follows from the fact that for any constant homogeneous transform

$$H = \begin{pmatrix} R_0 & \overline{x}_0 \\ \overline{0}^T & 1 \end{pmatrix},$$

$$J(gH) = \left[\left(\frac{\partial g}{\partial q_1} H(gH)^{-1} \right)^{\vee} \cdots \cdot \left(\frac{\partial g}{\partial q_{D(D+1)/2}} H(gH)^{-1} \right)^{\vee} \right].$$

Since $(gH)^{-1} = H^{-1}g^{-1}$, and $HH^{-1} = 1$, we have that $J(gH) = J(g)$.

The left invariance follows from the fact that

$$J(Hg) = \left[\left(H\frac{\partial g}{\partial q_1} g^{-1}H^{-1} \right)^{\vee} \cdots \cdot \left(H\frac{\partial g}{\partial q_{D(D+1)/2}} g^{-1}H^{-1} \right)^{\vee} \right]$$

where from [17] and [18] e know

$$\left(H\frac{\partial g}{\partial q_1} g^{-1}H^{-1} \right)^{\vee} = Ad_H \left(\frac{\partial g}{\partial q_1} g^{-1} \right)^{\vee}$$

where

$$Ad_H = \begin{pmatrix} R_0 & \hat{x}_0 R_0 \\ 0 & R_0 \end{pmatrix}$$

and $\hat{x}_0$ is the skew symmetric matrix such that $\hat{x}_0\overline{c} = \overline{x}_0 \times \overline{c}$ [17], [18]. Therefore,

$$J(Hg) = Ad_H J(g).$$

But since $\det(Ad_H) = 1$

$$\det(J(Hg)) = \det(J(g)).$$

The left and right invariance of $d\mu$ follows directly from that of $\det(J)$.

REFERENCES

[1] D. G. Alciatore and C.-C. D. Ng, "Determining manipulator workspace boundaries using the Monte Carlo method and least square segmentation," *ASME Robotics: Kinematics, Dyn., Contr.*, vol. 72, pp. 141–146, 1994.

[2] U. Basavaraj, and J. Duffy, "End-effector motion capabilities of Serial manipulators," *Int. J. Robot. Res.*, vol. 12, no. 2, Apr. 1993, pp. 132–145.

[3] D. Blackmore, M. C. Leu, "Analysis of swept volume via Lie groups and differential equations," *Int. J. Robot. Res.*, vol. 11, no. 6, pp. 516–537, Dec 1992.

[4] G. S. Chirikjian, "A binary paradigm for robotic manipulators," in *Proc. 1994 IEEE Int. Conf. Robotics and Automation*, San Diego, CA, May 1994.

[5] ——, "Kinematic synthesis of mechanisms and robotic manipulators with binary actuators," *ASME J. Mech. Design*, vol. 117, pp. 573–580, Dec. 1995.

[6] ——, "Fredholm integral equations on the Euclidean motion group," *Inverse Problems*, vol. 12, pp. 579–599, Oct. 1996.

[7] I. Ebert-Uphoff and G. S. Chirikjian, "Efficient workspace generation for binary manipulators with many actuators," *J. Robot. Syst.*, pp. 383–400, June 1995.

[8] ——, "Generation of discrete manipulator workspaces and work envelopes," in *Proc. IASTED Conf. Robotics and Manufacturing*, Cancun, Mexico, June 14–17, 1995.

[9] ——, "Inverse kinematics of discretely actuated hyper-redundant manipulators using workspace densities," in *Proc. 1996 IEEE Int. Conf. Robotics and Automation*, May 1996, pp. 139–145.

[10] H. Flanders, *Differential Forms with Applications to the Physical Sciences*. New York: Dover, 1989.

[11] P. Halmos, *Measure Theory*. Princeton, NJ: Van Nostrand, 1965.

[12] J. U. Korein, *A Geometric Investigation of Reach*. Cambridge, MA: MIT Press, 1985.

[13] A. Kumar and K. J. Waldron, "Numerical plotting of surfaces of positioning accuracy of manipulators," *Mech. Mach. Theory*, vol. 16, no. 4, pp. 361–368, 1980.

[14] S.-J. Kwon, Y. Youm, and K. C. Chung, "General algorithm for automatic generation of the workspace for n-link planar redundant manipulators," *ASME Trans.*, vol. 116, pp. 967–969, Sept. 1994.

[15] A. B. Kyatkin and G. S. Chirikjian, "An operational calculus for the Euclidean motion group," Tech. Rep. RMS 96-1, Johns Hopkins Univ., Baltimore, MD, Sept. 1996.

[16] D. K. Maslen and D. N. Rockmore, "Generalized FFTs—A survey of some recent results," *DIMACS Series Discrete Math. Theoretical Comp. Sci.*, vol. 28, pp. 183–237, 1997.

[17] J. M. McCarthy, *An Introduction to Theoretical Kinematics*. Cambridge, MA: MIT Press, 1991.

[18] R. M. Murray, Z. Li, S. S. Sastry, *A Mathematical Introduction to Robotic Manipulation*. Ann Arbor, MI: CRC, 1994.

[19] L. Nachbin, *The Haar Integral*. Princeton, NJ: Van Nostrand, Princeton, 1965.

[20] F. C. Park, "Robot sensor calibration: Solving $AX = XA$ on the Euclidean group," *IEEE Trans. Robot. Automat.*, vol 10, no. 5, pp. 717–721, Oct. 1994.

[21] F. C. Park and R. W. Brockett, "Kinematic dexterity of robotic mechanisms," *Int. J. Robot. Res.*, vol. 13, no. 1, Feb. 1994, pp. 1–15.

[22] D. L. Pieper, "The kinematics of manipulators under computer control," Ph.D. dissertation, Stanford University, Stanford, CA, Oct. 1968.

[23] J. Rastegar, and P. Deravi, "Methods to determine workspace, its subspaces with different numbers of configurations and all the possible configurations of a manipulator," *Mech. Mach. Theory*, vol. 22, no. 4, pp. 343–350, 1987.

[24] ——, "The effect of joint motion constraints on the workspace and number of configurations of manipulators," *Mech. Mach. Theory*, vol. 22, no. 5, pp. 401–409, 1987.

[25] B. Roth, J. Rastegar, and V. Scheinman, "On the design of computer controlled manipulators," First CISM-IFTMM Symp. on Theory and Practice of Robots and Manipulators, 1973, pp. 93–113.

[26] D. H. Sattinger and O. L. Weaver, *Lie Groups and Algebras with Applications to Physics, Geometry, and Mechanics*. New York: Springer-Verlag, 1986.

[27] D. Sen and T. S. Mruthyunjaya, "A discrete state perspective of manipulator workspaces," *Mech. Mach. Theory*, vol. 29, no. 4, pp. 591–605, 1994.

[28] M. Sugiura, *Unitary Representations and Harmonic Analysis*, 2nd ed. Amsterdam, The Netherlands: North-Holland, 1990.

[29] N. J. Vilenkin and A. U. Klimyk, *Representation of Lie Groups and Special Functions*. Dordrecht, The Netherlands: Kluwer, 1991, vols. 1–3.

[30] F. W. Warner, *Foundations of Differentiable Manifolds and Lie Groups*. New York: Springer-Verlag, 1983.

Gregory S. Chirikjian (M'93) received the B.S.E. degree in engineering mechanics, the M.S.E. degree in mechanical engineering, and the B.A. degree in mathematics, all from the Johns Hopkins University, Baltimore, MD, in 1988, and the Ph.D. degree from the California Institute of Technology, Pasadena, CA, in 1992.

Since the summer of 1992, he has been with the Department of Mechanical Engineering, Johns Hopkins University, where he is now an Associate Professor. His research interests include the kinematic analysis, motion planning, design, and implementation of "hyper-redundant," "metamorphic," and "binary" manipulators.

Imme Ebert-Uphoff received the degree Diplom der Techno-Mathematik in 1993 from the University of Karlsruhe, Germany, and the Ph.D. degree in 1997 from the Department of Mechanical Engineering, Johns Hopkins University, Baltimore, MD. Her thesis research focused on the development of discretely actuated, hyper-redundant manipulators with an emphasis on workspace generation and inverse kinematics. Further research interests include the study of parallel platform mechanisms and the development of simulation tools for that purpose.

She is currently a Postdoctoral Researcher at Laval University, Quebec City, P.Q., Canada.