David DeMers
Prediction Company
Santa Fe, NM 87501, USA

Kenneth Kreutz-Delgado
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, CA 92093, USA

Learning Global Properties of Nonredundant Kinematic Mappings

Abstract

The kinematic mapping $x = f(\theta)$ is generally many to one. For nonredundant manipulators, this means that there are a finite number of configurations (joint angles) that will place the end-effector at a target location in the workspace. These correspond to postures of the manipulator, and each configuration lies on a specific solution branch. It is shown that for certain classes of revolute joint regional manipulators (those with no joint limits and having almost everywhere a constant number of inverse solutions in the workspace), the input-output data can be analyzed by clustering methods in order to determine the number and location of the solution branches. As a practical consequence, the inverse kinematic mapping can be directly approximated by applying neural network or other learning-based methods to each branch separately.

1. Introduction

The main goal of the work presented in this paper is to develop methods of generating direct inverse kinematics functions for nonredundant manipulators. The kinematics mapping has a generic topological structure such that the workspace can be partitioned into specific well-defined regions that are invertible. We describe the topology of the kinematics mapping and present heuristic algorithms, which, when applied to data from a manipulator, can identify the invertible regions of the workspace and the corresponding jointspace solution branches of each such invertible region.

1.1. The Inverse Kinematics Problem

The forward kinematic function of a serial-chain manipulator $f : \Theta^n \rightarrow \mathcal{W}, \mathcal{W} = f(\Theta^n)$ maps a set of n joint parameters from the n-dimensional manipulator configuration space Θ^n to the m-dimensional workspace $\mathcal{W}$, $m \leq n$:

$$x = f(\theta),\ x \in \mathcal{W}.$$

The inverse kinematics problem in robot kinematics is to solve the nonlinear inverse problem:

$$\text{Given } x \in \mathcal{W},\ \text{find } \theta \text{ such that } f(\theta) = x. \qquad \text{(IKP)}$$

In this paper it is assumed that x is in the reachable workspace $\mathcal{W}$, and thus that a solution exists.

The forward kinematic mapping $f(\theta)$ is many to one; therefore, a solution to problem (IKP) will not be unique. There are both global and local manifestations of this nonuniqueness; generically, the preimage of a particular x consists of a finite number of distinct sets of configurations, each set being a manifold in the configuration space[1] of dimensionality $n - m$ (Burdick 1988).

In this paper, we consider only nonredundant manipulators for which $n = m$, so that the existence of multiple solutions arises from the fact that the mapping is globally finitely many to one, although locally one to one.[2] Issues pertaining to learning inverse kinematics for redundant manipulators are discussed in DeMers (1993) and DeMers and Kreutz-Delgado (1996). In arriving at a unique solution to $f^{-1}(x)$ for nonredundant manipulators, a choice of one of the finite number of preimage points that solve $f(\theta) = x$ (i.e., which posture) must be made.

The exact number of inverse solution branches c generally depends on several factors, including the task space topology and geometry, the specific parameters that prescribe the arm geometry, and the location of the end-effector in the task space. However, the number is always finite, and relatively small. It is known that there are a maximum of 16 inverse solution branches for any full spatial manipulator (operating in SE(3)) (Raghavan and Roth 1989) and a maximum of 4 inverse solutions for regional 3R manipulators (operating in $\mathbb{R}^3$) (Pieper 1968). Furthermore, for 3R regional manipulators, the number of solutions is either 2 or 4 over all regular values (i.e., nonsingular locations) in the workspace (Hsu and Kohli 1987; Spanos and Kohli 1985).

The International Journal of Robotics Research,
Vol. 17, No. 5, May 1998, pp. 547–560,

1. Also often called the joint angle space. In this paper, the term configuration space is generally used.
2. Specifically, it is assumed that $f(\cdot)$ is a local diffeomorphism.

In this paper, for clarity, only manipulators such that the number of solution branches is almost everywhere constant in the workspace will be considered. These are the Type 1 manipulators discussed in Burdick (1988). However, the results generalize to all nonredundant 3R manipulators by the methods used to generalize to redundant manipulators given in DeMers (1993). Further theoretical work might possibly extend the lemmas discussed below to broader classes of manipulators.

This paper is primarily empirical, although heavily motivated by theoretical results from differential topology, which are used to give a precise and formal definition of "solution branch." This paper relies on theorems from topology and their application to robot inverse kinematics as developed in Burdick (1988), Baker (1990), and Wenger (1991). Generic topological characteristics of the inverse kinematic structure motivate unsupervised learning techniques that identify and label solution branches. These methods are applied to several manipulators. Given such solution branch identification, direct inverse functions can be estimated. In this paper, the inverse mappings to (IKP) for all possible solution branches are learned.

In Section 2, necessary background in topology and kinematics is presented. In Section 3, it is shown that the solution branches can be identified from the input-output data (joint angle, workspace location pairs) of a robot in the absence of knowledge of the form of the forward kinematics mapping. We then apply the methods to synthetic data from four hypothetical 3R manipulators and measured data from the PUMA 560, and show that we can obtain a good approximation to the inverse solution branches.

It is clearly not necessary to use data-driven learning methods if the forward kinematics is known or if the inverse solutions are available in closed form, as for 3R manipulators. However, this paper shows that even in the absence of knowledge of the forward kinematics, the inverse can be approximated and solution branches identified from a sample of input-output data by exploiting the topological characteristics of the kinematics mapping.

2. Topology of the Kinematics Map

The workspace of a manipulator consists of disjoint regions that are separated by the image of singular configurations (Burdick 1988). Each such region (by definition) is a maximal connected set that contains no workspace locations that can be reached in a singular configuration. The inverse image of one such region consists of c disjoint regions in the configuration space. Any target end-effector location in the workspace region can be reached in one of c different postures. For example, the (wristless) Puma 560 positions its end-effector in a particular location in one of four postures, depending on whether the elbow is up or down and whether the shoulder is on the right or the left.

Borrel (1986) defines the aspects of the configuration space to be the maximally connected regions of the joint space Θ^n such that the Jacobian matrix remains full rank. The aspects are a finite number of disjoint open sets whose closure is the complete jointspace.

Let $\mathcal{A}_i$ be an aspect of Θ^n. The characteristic surfaces of $\mathcal{A}_i$, $SC(\mathcal{A}_i)$, are defined to be the preimage set in $\mathcal{A}_i$ of the image of the boundary points, $B\mathcal{A}$, of $\mathcal{A}_i$ (Wenger 1991, 1992). Thus, $SC(\mathcal{A}_i) = f^{-1}(f(B\mathcal{A}_i))$. The characteristic surfaces partition each aspect $\mathcal{A}_i$ into a finite number of basic components. A single aspect consists of the union of its basic components plus the characteristic surfaces. The images of the basic components in the workspace are called basic regions.

The basic components form a disjoint partition of the aspects, and the aspects form a disjoint partition of the configuration space. Thus, a point in the configuration space will be either in a unique aspect and a unique basic component or part of one of the characteristic surfaces. All points in the workspace have no more than one unique inverse solution in any basic component (Wenger 1991, 1992; Borrel 1986). Therefore, any change of posture must involve passing through either a singularity or a characteristic surface.

Burdick (1988) developed a taxonomy of singular surfaces that can exist in the configuration space, and called these critical point surfaces (CPS). Type 1 manipulators must pass through a CPS to change posture. For Type 1 manipulators, the characteristic surfaces and CPS are equivalent, and thus the aspects are the basic components. For other manipulators, the characteristic surfaces may not be CPS. The CPS are also referred to as Jacobian surfaces by Hsu and Kohli (1987); this is because they are connected configuration space surfaces such that for all configurations on the surface, the Jacobian of the forward kinematics mapping loses rank. That is, the manipulator is at a singularity.

A solution branch of (IKP) over a basic region is defined to be a single basic component. For nonredundant 3R regional manipulators, the characteristic surfaces partition Θ^n into c disjoint basic components, or solution branches. Denote these $\mathcal{B}_i$, indexed by solution branch number $i \in \{1, \ldots, c\}$. Thus, $\Theta^n = \overline{\bigcup_{i=1}^{c} \mathcal{B}_i}$, where $\mathcal{B}_i \bigcap \mathcal{B}_j = \emptyset$ for $i \neq j$,[3] with the overbar denoting set closure. The image of each basic component $\mathcal{B}_i$ under f is a connected basic region in the workspace, $\mathcal{W}_i = f(\mathcal{B}_i)$, and the entire workspace, $\mathcal{W} = \overline{\bigcup_{i=1}^{c} \mathcal{W}_i}$. Let $\hat{f}_i$ denote the kinematics mapping restricted to $\mathcal{B}_i$; $\hat{f}_i : \mathcal{B}_i \rightarrow \mathcal{W}_i$. The function $\hat{f}_i$ is invertible, and hence the problem of learning $\hat{f}_i$ or its inverse is well defined.

For the class of manipulators considered in this paper, every regular value $x \in \mathcal{R}$ has exactly c inverse solutions.

3. The limit points of the $\mathcal{B}_i$ are the characteristic surfaces; thus, more precisely, $\Theta^n = \bigcup_{i=1}^{c} \mathcal{B}_i + \bigcup_{i=1}^{c} SC(\mathcal{A}_i)$.

Thus, the number of solutions to (IKP) is almost everywhere constant over $\mathcal{W}$. For all classes of nonredundant regional manipulators, the property that $\mathcal{W}_i = \mathcal{W}$, $\forall i$, is sufficient for c to be almost everywhere constant in the workspace $\mathcal{W}$. Thus the CPS, or Jacobian surfaces, are workspace boundaries (Hsu and Kohli 1987). This means that the only way for the manipulator to change posture is to move to the workspace boundary singularity. Therefore, the properties that (i) c is almost everywhere constant and (ii) there are no joint limits on revolute joints are necessary and sufficient for $\mathcal{W}_i = \mathcal{W}, \forall i = 1, \ldots, c$ (and sufficient for the manipulator to be Type 1).

For a Type 1 3R regional manipulator, the kinematics map is $f : T^3 \to \mathcal{W} \subset \mathbb{R}^3$. The solution branches for a particular x are the basic components $\mathcal{B}_i$. These are connected areas of configuration space separated by the characteristic surfaces (here, the CPS). The basic components correspond to the notion of posture regions of the manipulator.

3. Identifying the Topological Components

In this section, methods are developed that process measured (θ, x) data and partition the configuration space into the basic components $\mathcal{B}_i$.

Given a set of sample data, consisting of (θ, x) pairs, the solution branches (basic components) are identified by examining the inverse of a set of regions in the workspace as follows:

- Choose an ordered set of M query points (neighborhood centers) $\{x_q \in \mathcal{W}, q = 1, \ldots, M\}$ in the workspace $\mathcal{W}$.
- For each query point x_q, find all x in the data sample that lie within some neighborhood of x_q, and retrieve the associated θ.
- Partition the θ points into c clusters.
- Label the clusters; that is, identify in which component (solution branch) of Θ^n each cluster belongs.
- Given labeled data, use supervised learning to build a solution branch classifier for Θ^n, and direct inverse kinematics functions for each branch.

In the examples developed in this paper, data samples are generated by sampling from a uniform distribution in configuration space. The amount of data and the method of acquiring or generating it will affect the quality and accuracy of the results.

The query points, labeled as x_q, serve to locate neighborhoods in the workspace containing sample data points x. They should be chosen in the workspace $\mathcal{W}$ such that

- each query point has sufficiently many data sample points x, near it; and
- query point x_{q+1} is near query point x_q.

In this paper, query points are selected by fitting an M node 1D Kohonen network to the entire set of x points, where each of the nodes corresponds to a query point $x_q, q = 1, \ldots, M$. If there are M nodes in the network and N data points ($N > M$), the Kohonen learning algorithm adjusts their location in the data space such that each node is the nearest node to approximately N/M data points. In addition, a topological ordering on the nodes is enforced that locates node $q+1$ near node q in the data space. The precise details of this algorithm are beyond the scope of this paper; specific implementation details can be found in Kohli and Hsu (1987) and DeMers (1993).

The nodes of the trained Kohonen network are distributed throughout Θ^n with the density of the data, and the 1D topology provides a smooth path in the workspace such that node q is relatively close to node $q + 1$. This is a data-driven method of query point selection, guaranteeing that query points will be in areas of the workspace for which data samples exist, and that query point x_{q+1} is near query point x_q. In the limit that $M, N \to \infty$, the sequence of query points will follow a space-filling curve through the data.

The topology of Type 1 manipulators is such that the preimage of a neighborhood in the workspace consists of a finite number of neighborhoods in the configuration space. All of the pairs in the sample database with their x component lying inside a small neighborhood of x_q are retrieved along with their associated preimage points. The preimage points generically fall into c groups, c being almost everywhere constant by assumption, which are separable and identifiable by the use of clustering methods. Figure 1 illustrates the query point neighborhood and its inverse image. The retrieved points will fall within the boundaries of the inverse neighborhoods.

An example from one of the experiments reported in Section 4 is shown in Figure 2 and Figure 3 shows the inverse clusters for workspace points that position the manipulator (a wristless Puma 560) in a neighborhood of query point $x_q = (0, 30, 40)$. These clusters correspond to the various physical postures of the arm shown in Figure 4.

The distance between points is used for clustering; however, choice of a distance metric can be problematic. In the examples analyzed in this paper, the configuration space is a product space of three angles, which has the topology of a 3-torus, T^3. Thus, the data lie on a Riemannian manifold, and the distance between two joint vectors is a measure of their (dis)similarity. Given a collection of joint angle vectors, let us represent the ith joint vector as $\theta_i = (\theta_{i,1}, \theta_{i,2}, \theta_{i,3})$, where each $\theta_{i,k} \in [-\pi, \pi)$. A well-defined metric on the torus to compute distance between θ_i and θ_j is $\mathcal{D}(\theta_i, \theta_j)$, where

$$\mathcal{D}^2(\theta_i, \theta_j) = \sum_{l=1}^{3} \min(|\theta_{i,l} - \theta_{j,l}|, 2\pi - |\theta_{i,l} - \theta_{j,l}|)^2.$$

For spaces defined by ball-and-socket or screw joints, a metric on configurations should be chosen carefully.

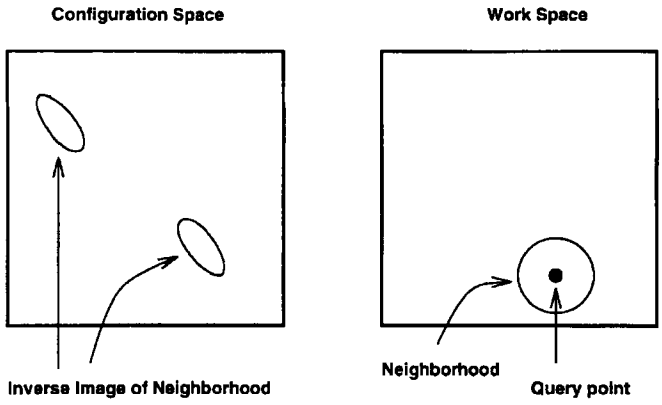

Fig. 1. Query point neighborhood and its inverse. For a single query point x_d, all points in the workspace inside of a neighborhood of radius ϵ are retrieved. The associated configuration space points will fall within the inverse image of the boundary of the ϵ-ball. Thus, by construction, separable groups of configuration space points are obtained.

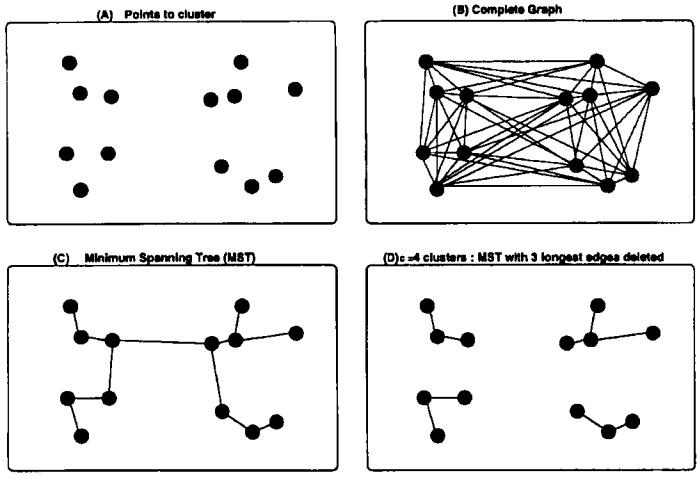

Fig. 2. Clustering via minimum spanning tree, $c = 4$. Points in the preimage of a query point considered as nodes of a graph; the edges have cost equal to the distance between the two points connected by the edge. (a) Thirteen points in the preimage of a query point, (b) the complete graph (with some edges left out for clarity of presentation), (c) the minimum spanning tree, (d) four clusters obtained by deleting the three longest edges of the minimum spanning tree.

Single-linkage clustering will optimize

$$J \triangleq \sum_{i,j=1}^{c} d(C_i, C_j)$$

(see Duda and Hart 1973). Many implementations of single-linkage clustering exist. We chose to use the dual problem of computing a minimum spanning tree (MST) through a graph in which the nodes are the points and the arcs have cost equal to the distance between the two points connected.

The MST is a single cluster containing all of the data points. After the MST has been constructed, deletion of $c - 1$ edges partitions the graph into c components.

If c is unknown, heuristics will be needed to determine the number of clusters actually appearing. Zahn (1971) suggested removing "inconsistent edges." Let ratio(e_{ij}) be the ratio of the length of an edge with the average of the lengths of all of the other edges incident on either endpoint node. This is computed as

$$|e_{ij}| \Big/ \frac{1}{\deg(\theta_i) + \deg(\theta_j) - 1} \left(\sum_{k \neq j \text{ s.t. } e_{ik} \in MST} |e_{ik}| + \sum_{k \neq i \text{ s.t. } e_{jk} \in MST} |e_{jk}| \right)$$

(where $\deg(\theta_i)$ indicates the number of edges incident on the node of the MST that corresponds to θ_i). A high value for *ratio*(e_{ij}) indicates that the edge probably connects distinct clusters and is inconsistent, and thus can be deleted, whereas a low value indicates that the edge is more likely an intracluster link and, thus, should not be deleted.

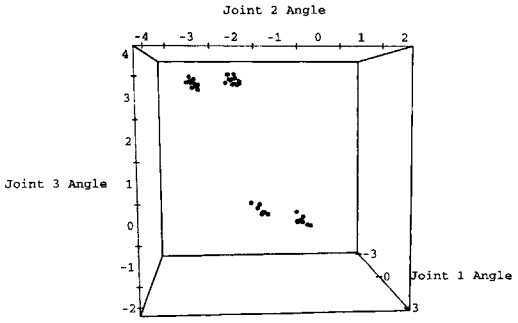

Fig. 3. The configuration space points in the data sample corresponding to data for which the end-effector of the Puma 560 is within 10 cm of location (0, 30, 40) in the workspace.

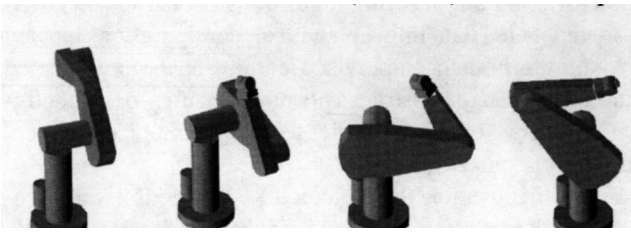

Fig. 4. The four physical solutions that position the end-effector at (0, 30, 40).

3.1. *Possible Clustering Problems*

Under some circumstances the inverse set for a particular query point neighborhood may not consist of easily separable clusters. For example, when the query point is near a singularity, inverse clusters will join together. In this case, the inverse image of the ball in the workspace around the query point, as illustrated in Figure 1, is a set of fewer than c highly distorted balls in the configuration space. The amount of distortion depends on the inverse Jacobian around the query point.

Another problem may occur if the θ set fails to include at least one member from each of the inverse solutions. In this case, the clustering algorithm should report the fact that there are fewer clusters than expected.

3.2. *Classifying the Samples—Merging Clusters*

The result of the sampling and clustering, after stepping through all M query points $x_q, q = 1, \ldots, M$, is a sequence of $\overline{M}, \overline{M} \leq M$, sets of c clusters. By construction, since query point x_{q+1} is near query point x_q, clusters in neighboring sets in the sequence should also be nearby. Let $s_q = \{C_1^q, \ldots, C_c^q\}, i = 1, \ldots, \overline{M}$ be sets, where s_q contains the c clusters C_j^q obtained for query point x_q. For each q, each of the c configuration space clusters lies in one of the c solution branches, and no two are in the same branch. We want to assign each C_j^q to the proper c solution branch. This is equivalent to assigning each C_j^q to the correct global categories, where a category indicates a particular posture (i.e., solution branch).

- These categories (solution branches) exist in a global and unambiguous sense for the class of admissible manipulators, as discussed in Section 2.

There are a combinatorial number ($\mathcal{O}(c!^{\overline{M}})$) of possible assignments, thus some approximation algorithm using reasonable heuristics must be used, and a reasonable optimization criterion or objective function must be developed.

The goal at this step is to maximize the number of globally correct assignments. The quality of an assignment can be estimated as the sum of all of the within-category distances, since the solution branches are connected, and, in general, nearby points in the configuration space will be in the same solution branch. The elements of each set s_q must be labeled to be consistent with the global categories (solution branches). Given a labeling of all $\overline{M}$ of the sets, define

$$\mathcal{H} = \Sigma_{q=1}^{c} \Sigma_{k=1}^{\overline{M}} \Sigma_{j=k}^{\overline{M}} d(C_k^q, C_j^q).$$

We want to obtain the labeling that minimizes $\mathcal{H}$, subject to the restriction that for each s_q, each C_j^q must have a unique label.

There are a number of approximation algorithms to use. Because it is known that $f(\cdot)$ is continuous, it can be expected that C_j^q will generally move to a nearby location C_j^{q+1} as x_q is moved to the neighboring end-effector location x_{q+1}. Thus, a greedy algorithm is appropriate. Clusters can be labeled on-line, since only previously labeled sets of data need to be examined in order to label the current set of clusters.

Let $d(i, j)$ be the distance from C_i^k to the nearest member of category j. If all C_j^q are labeled for all $q < k$, a locally optimal labeling would select the c distances $d(i, j)$ such that each i and each j appear exactly once, and minimize the total cost of the distances, $\sum_{i=1}^{c} d(i, j)$. This results in algorithm LABEL.

ALGORITHM LABEL: On-Line Labeling, Query Point q

1. For each of the c clusters $C_i^q, i = 1, \ldots, c$
2. For each of the classes $j = 1, \ldots, c$
3. Compute $d(i, j)$
4. Minimize $\sum_{i=1}^{c} d(i, j)$ such that each i and j appear exactly once in the sum.

For each q, the naive implementation of step 4 is $\mathcal{O}(c!)$. For $c \leq 4$, as in the case of regional manipulators, they can be examined exhaustively. For the general case of a spatial manipulator in which $c \leq 16$, exhaustive search may be prohibitive. However, the problem can be reformulated as $N - 1$ instances of minimum-weight perfect bipartite matching, for which an efficient algorithm is known. The now-labeled clusters C_i^q are added to the set of labeled data, and algorithm

LABEL is then applied to s_{q+1}. Algorithm LABEL is an on-line algorithm in the sense that it depends only on previously labeled data.

Algorithm LABEL may make incorrect assignments when the query points are too far apart (and thus the continuity principle is violated) or when the query point is near a singularity. Experimentally, the use of algorithm LABEL occasionally can create and propagate erroneous assignments. These can be identified and eliminated by comparing the costs of the best with the next best labeling; when erroneous assignments were made, the value of the sum in step 4 for the next best assignment was close in value to that of the best, whereas if the cost of the best assignment was less than half of the second best cost, the labeling produced (for our experiments) was always correct. These cases occurred less than 3% of the time for the experiments given in Section 4.

Figures 5 and 6 illustrate algorithm LABEL for the four posture category case. In Figure 5, the numbered nodes represent the four inverse clusters obtained for the query point of the same number. We assume that those for query points 1 and 2 have been assigned to a posture class, and the four clusters from query point 3 are now to be assigned a posture label. The on-line algorithm finds the distances from each cluster to the nearest member of each posture class, illustrated in Figure 6, and labels the clusters such that the sum of distances to the nearest member of its own class is minimized, subject to the constraint that no two clusters may be in the same class.

4. Learning Inverse Kinematics

The work reported in this paper is motivated by recent efforts to use nonlinear function approximation methods on measured joint angle and end-effector position data in order to learn the inverse kinematic function; that is, to estimate $f^{-1} : (x) \mapsto \theta$ from (x, θ) data samples. Such direct inverse kinematics approaches using neural networks as the function approximator can be found in Kuperstein (1987, 1988, 1991), Barhen, Gulati, and Zak (1989), Martinetz, Ritter, and Schulten (1990), and Ritter, Martinetz, and Schulten (1989).

Algorithms that can result in the learning of direct inverse functions are appealing; a great advantage of these algorithms is their speed and minimal computational cost at runtime, which greatly facilitate real-time applications. However, a basic and serious difficulty with these learning methods, which attempt to learn the direct mapping from position to joint angles, is that they fail when multiple inverse solutions exist. Mean-squared error approximations produce an average over the training data. Such an average is not guaranteed to be a valid solution when the inverse is a nonconvex set (Jordan and Rumelhart 1990). Some means of choosing between multiple solutions is needed. In practice, the neural network approaches to direct inverse kinematics allow only a single-solution possibility that restricts the manipulator to a single posture.

The algorithms developed in this paper allow one to solve the full inverse problem by first partitioning the data into regions that are invertible. Once solution branches are identified, and input-output data are separated along these branches, the inverse kinematic mapping can be directly approximated on each branch separately using any of the methods that apply to the one-to-one case, such as those of Kuperstein (1991) and Martinetz, Ritter, and Schulten (1990).

4.1. *Building a Solution Branch Classifier*

The clustering and labeling algorithms bootstrap the previously unlabeled data into data with all configurations labeled as to solution branch. Supervised learning on the now-labeled data can be used to construct a posture classifier for all configurations. Any suitable classifier technique can be employed. Call this classifier $\mathbb{B}(\theta)$.

Figure 7 illustrates the architecture of a feedforward neural network to classify solution branches for the PUMA 560. The network can be trained to identify in which of the four solution branches a configuration θ is. The network shown has three inputs (for joint angles) and four outputs (one for each of the four posture classes). Given the labeled data, the network can be trained to classify the 3D configuration space vectors into one of the posture classes. The architecture shown in Figure 7 is used in this paper to approximate $\mathbb{B}(\theta)$, although the authors have also used nearest neighbor methods. Complete implementation details and design decisions can be found in DeMers (1993).

This classifier can be used to estimate both the posture class to which a configuration belongs and the location of the class boundaries, which are the singularities of this robot. For example, many configurations drawn from a uniform distribution over the configuration space can be presented to the network, and if the classification unit with maximum activation is not very strong (or equivalently, if multiple units have nontrivial activation), one can conclude that the network's "confidence" in its classification is low. Low-confidence configuration space points are the sets of joint angles at which the manipulator is near a singularity. Using $\mathbb{B}(\theta)$ to select low-confidence points allows one to experimentally identify and map out the critical point surfaces, as shown in the next section.

5. Application to 3R Regional Manipulators

In this section, the methods of Section 2 are applied to data generated from four 2-solution Type 1 3R regional manipulators and to the wristless Puma 560 (with no joint limits), a 4-solution Type 1 manipulator. The 2-solution manipulators are adapted from Burdick (1988). The CPS have a qualitatively differing structure for each of the manipulators

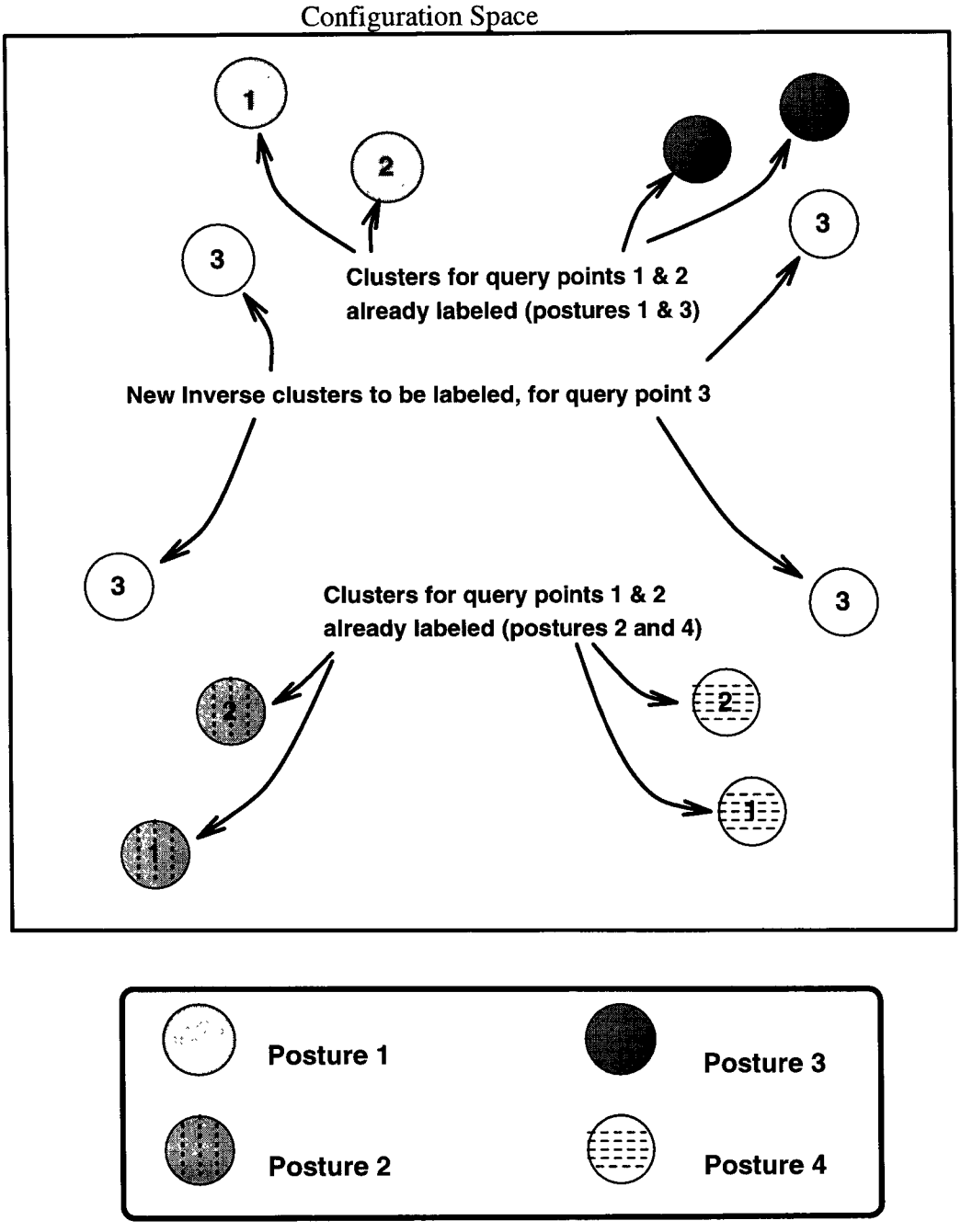

Fig. 5. Labeling clusters on-line for the four posture category case. Clusters from query points 1 and 2 have been previously assigned to posture classes (equivalently, have been assigned posture labels). Now, the four clusters for the next query point $q = 3$ are identified via Algorithm 2 and are to be labeled on-line.

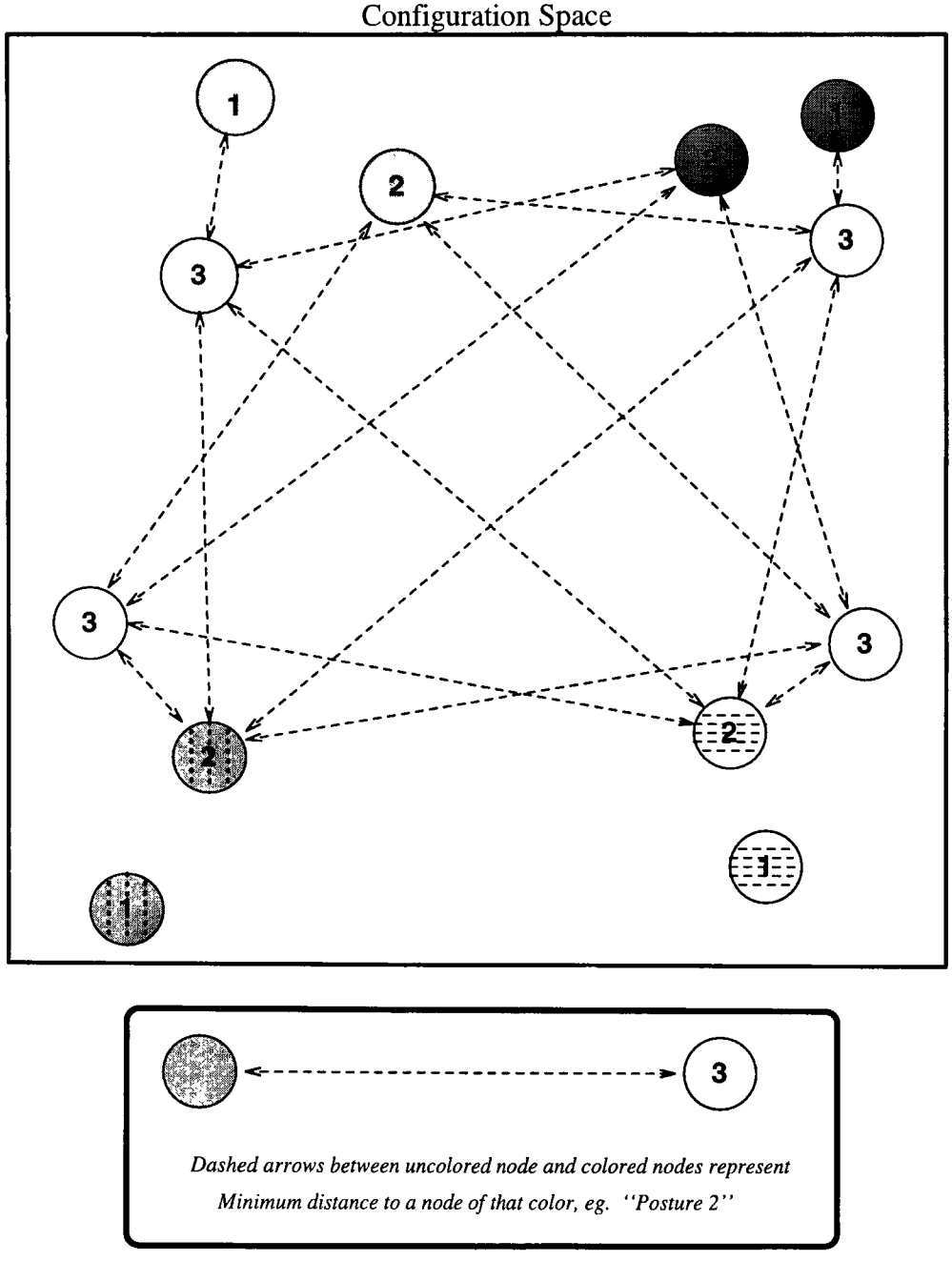

Fig. 6. Labeling clusters on-line for the four posture category case. The clusters are assigned to posture classes such that the sum of the distances from each to the nearest already-assigned member of the same class is minimized, and such that each is assigned to a unique one of the four classes.

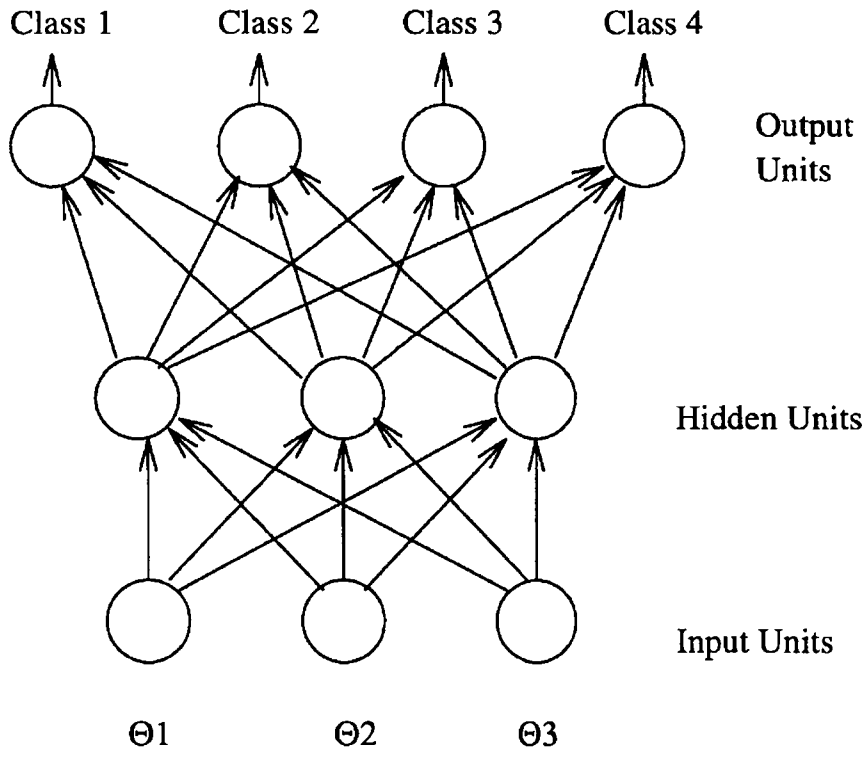

Fig. 7. Neural network classifier. This feedforward neural network, when trained on the bootstrapped labeled data, estimates the class conditional probability for an input configuration space vector.

considered. The methods described in this paper are shown to be able to decompose the configuration spaces into the disjoint basic components mapping one to one and onto their images. The use of topological understanding to guide the application of the algorithms is discussed. A more detailed description and discussion of the applications presented here can be found in DeMers (1993).

5.1. Two-Solution Manipulators

First, we show a generic 2-solution manipulator that is representative of the class of 2-solution manipulators in that the CPS are relatively smooth and well behaved. Next, we show two manipulators on either side of a CPS bifurcation (referred to as pre- and postbifurcation manipulators) that cause qualitative changes to the nature of the CPS, although the two manipulators differ only by a slight amount in a single one of their DH parameters. Finally, we show another 2-solution manipulator ("Isola") with substantially different CPS. These examples serve to illustrate the fact that the methods are reasonably effective in partitioning the configuration space for a wide range of 2-solution 3R manipulators.

In Figures 8 to 16, the true CPS are shown, along with the CPS as estimated by constructing $\mathbb{B}(\theta)$ in the manner described in Section 3.5, for each of these manipulators. The volume of the configuration space that is misclassified ranges from about 1% in the case of the Puma to 9% in the case of the near-bifurcation manipulator. The misclassified points can generally be identified as being near the singularity. Detailed analysis of issues surrounding the classification error can be found in DeMers (1993).

The data are generated by sampling from a uniform distribution in configuration space. Since joint 1 is revolute and unlimited, only data that lie in a 3^o vertical slice of the workspace is retained. This vertical slice is a complete cross-section of the workspace; as joint angle θ_1 is rotated, it is swept around the Z-axis of the base frame.

The configuration space is projected to the θ_2-θ_3 2-torus. The singularities, and thus the CPS, of 3R manipulators do not depend on θ_1. The configuration space shown in the figures is the θ_2-θ_3 2-torus "sliced" along its generators and "unfolded" into a square; note that the opposite edges of the square are identified. For all of these experiments, 6,000 data samples were taken in the vertical slice. This yields equivalent sampling density to 720,000 samples in the entire workspace.

Within each slice, 300 query points are chosen, and the preimage data are clustered and labeled. A neural network classifier is then trained on this labeled data to obtain an estimate of $\mathbb{B}(\theta)$. The two classification output units of the network range from 0 to 1 in "activation" value. A new configuration space point (set of joint angles), when presented as

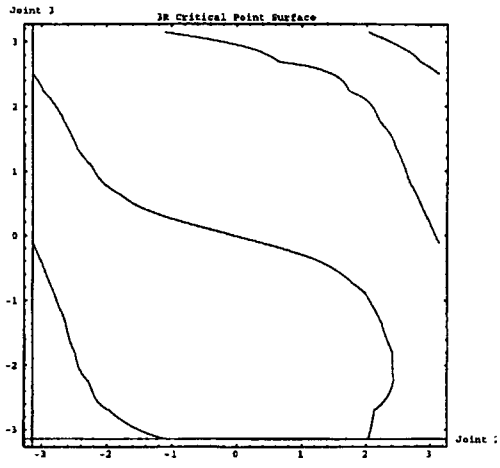

Fig. 8. Generic 2-solution. The true CPS in the configuration space (shown projected to the θ_2-θ_3 plane), computed by Mathematica from the determinant of the Jacobian.

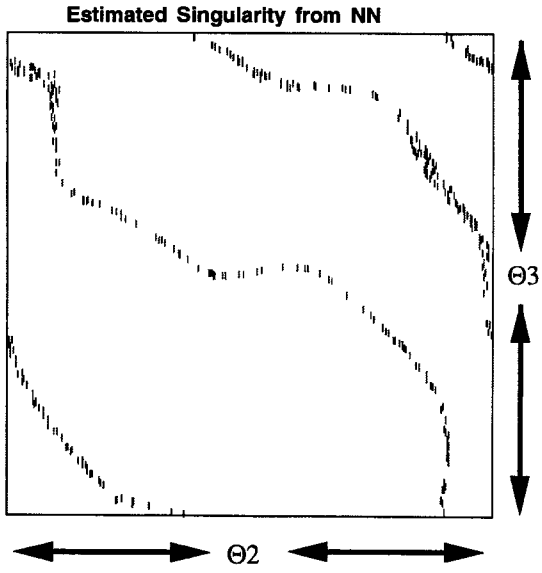

Fig. 9. Generic 2-solution. The estimated CPS in the configuration space (shown projected to the θ_2-θ_3 plane), generated by the neural network classifier with threshold of 0.6.

input to the neural network, will cause these output units to activate; the configuration is classified as to solution branch based on which unit has maximum activation. $\mathbb{B}(\theta)$ is used to classify new configuration space points and to produce an estimate of the confidence in the classification, which is then used to estimate the boundaries (i.e., CPS) between posture classes. In addition, $\mathbb{B}(\theta)$ is used to estimate the volume of configuration space for which the confidence in the classification falls below any particular threshold. The volume is estimated by computing the frequency of classification by $\mathbb{B}(\theta)$, with respect to a confidence threshold, of new samples drawn from a uniform distribution in configuration space.

5.1.1. *The Generic 2-Solution Manipulator*

The DH parameters of this manipulator are $\alpha_1 = 60°, \alpha_2 = -90°, a_1 = 1.3, a_2 = 1.7, a_3 = 1.6, d_2 = 0.4$, and $d_3 = -0.2$. Our methods identified $c = 2$. Figure 8 shows the true CPS separating the two solution branches. Figure 9 shows the approximate estimated CPS, computed from $\mathbb{B}(\theta)$. The points along the surface shown in Figure 9 are low-confidence points for which the maximum activation of any of the output units of the neural network classifier did not exceed 0.6. The estimated CPS are quite close to the actual, with a few minor deviations primarily due to random sampling effects.

Configuration space points can be classified according to which output unit is maximally active. However, they can also be classified "uncertain" if this maximum activation is below a threshold. Using the threshold of 0.6 as in Figure 9, 91% of the data points were classified correctly and 4% were classified as uncertain, with the remaining 5% classified incorrectly. This is typical for 2-solution manipulators.

5.1.2. *Pre- and Postbifurcation 2-Solution Manipulators*

The DH parameters for both of these have $\alpha_1 = 50°, \alpha_2 = -30°, a_1 = 1.3, a_2 = 1.7, d_2 = -0.3$, and $d_3 = 0.4$. A bifurcation in the nature of the CPS occurs as a_3 is changed from 0.8 to 0.9. The clustering algorithm correctly identified the two solutions. Figure 10 shows the CPS for the prebifurcation manipulator with $a_3 = 0.8$, and Figure 12 shows the CPS for the postbifurcation manipulator with $a_3 = 0.9$. The estimated singularities are shown in Figures 11 and 13 for a confidence threshold of 0.6. Misclassification occurred for 5% of the points in the case of the prebifurcation manipulator and for 7% of the points in the case of the postbifurcation manipulator.

It can be seen that the basic structure of the prebifurcation manipulator is essentially recovered by application of our algorithms. However, the postbifurcation CPS are qualitatively erroneous. A closer inspection of the data and sampling reveals that the region in the upper right part of the illustrations is undersampled; either no query point was located near the image of these CPS or the configuration space data could not be separated successfully (i.e., heuristics indicated the presence of only a single cluster). The use of more sophisticated heuristics, such as resampling near and along estimated singularity surfaces, should lead to iterative refinement of the estimate.

5.1.3. *The "Isola" Manipulator*

This manipulator has DH parameters of $\alpha_1 = 60°, \alpha_2 = -45°, a_1 = 1.1, a_2 = 1.0, a_3 = 0.98, d_2 = 0.3$, and $d_3 =$

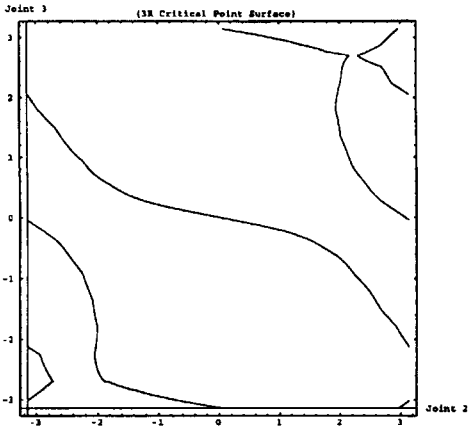

Fig. 10. Prebifurcation, 2-solution. The true CPS in the configuration space (shown projected to the θ_2-θ_3 plane), computed by Mathematica from the determinant of the Jacobian.

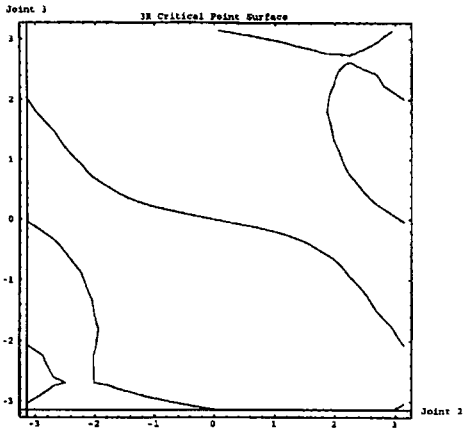

Fig. 12. Postbifurcation, 2-solution. The true CPS in the configuration space (shown projected to the θ_2-θ_3 plane), computed by Mathematica from the determinant of the Jacobian.

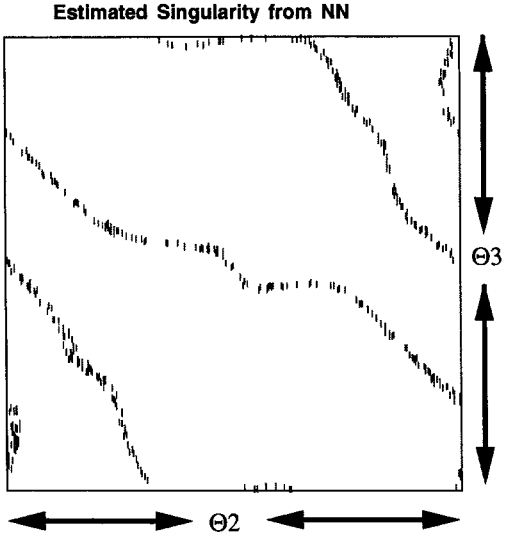

Fig. 11. Prebifurcation, 2-solution. The estimated CPS in the configuration space (shown projected to the θ_2-θ_3 plane), generated by the neural network classifier with threshold of 0.6.

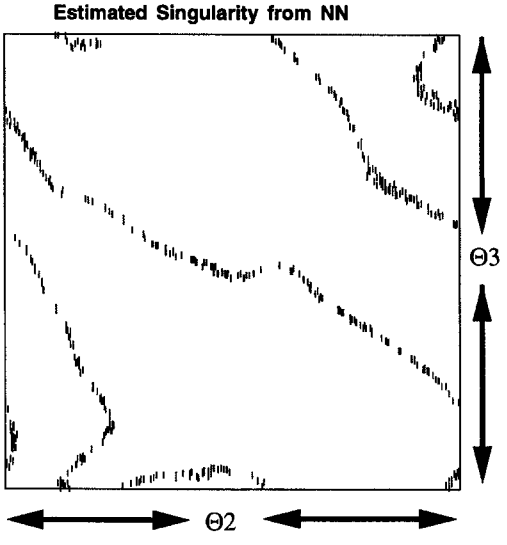

Fig. 13. Postbifurcation, 2-solution. The estimated CPS in the configuration space (shown projected to the θ_2-θ_3 plane), generated by the neural network classifier with threshold of 0.6.

0.3. It is near a bifurcation point in DH-parameter space where an "isolated" singularity appears (see Burdick 1988). Our methods correctly identified $c = 2$. As can be seen, in Figures 14 and 15, the CPS are substantially different in form from the other manipulators, yet the algorithms again found a good estimate of their location. Note that near the sharp corner, the region of low confidence is quite broad, but in general the singularity is crisply identified. The error rate in classification is quite low—3% misclassified and 2.5% uncertain.

5.2. *The Wristless Puma 560 with No Joint Limits*

The Puma 560 is a 6R spatial manipulator whose last three joint axes intersect; thus, it has a "separable wrist." The wristless Puma with no joint limits is a nongeneric Type 1 3R regional manipulator with two extra branch singularities (see Burdick 1988). Assuming no physical limits on the range of each joint angle, there are four disjoint regions in the configuration space of the Puma, each of which maps onto the entire reachable workspace. The specific DH parameters used are $a_2 = 0.4318$ m, $a_3 = -0.02032$ m, $\alpha_1 = -90^o$, $\alpha_3 = 90^o$, $d_2 = 0.148$ m, $d_3 = 0.4891$ m.

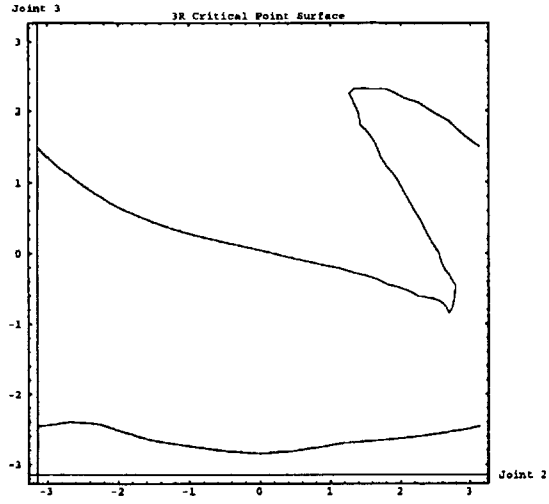

Fig. 14. Other 2-solution. The true CPS in the configuration space (shown projected to the θ_2-θ_3 plane), computed by Mathematica from the determinant of the Jacobian.

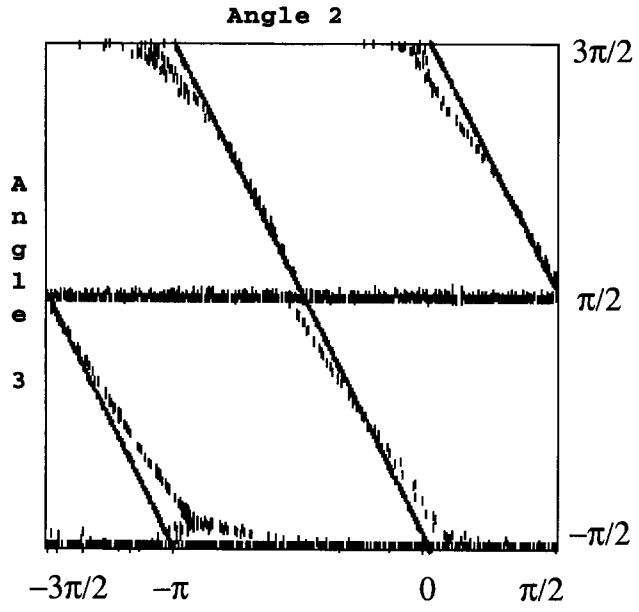

Fig. 16. Puma 560. The actual and estimated singularities in the configuration space of the wristless Puma with no joint limits. The estimates are produced from a neural network classifier trained on the bootstrapped labeled data. These are 2000 configuration space points for which the maximum activation of any of the four class nodes was less than 0.8.

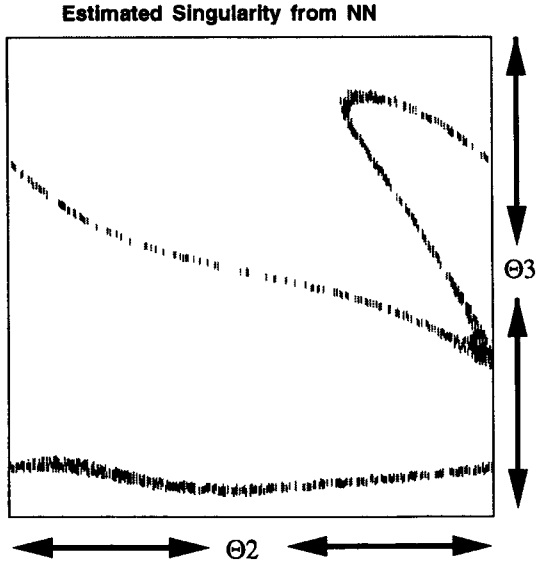

Fig. 15. Other 2-solution. The estimated CPS in the configuration space (shown projected to the θ_2-θ_3 plane), generated by the neural network classifier with threshold of 0.6.

Figure 3 shows an example of data points in the configuration space of the wristless Puma 560 that all map to the same neighborhood of query point $x_q = (0, 30, 40)$ in the workspace. The clusters in the configuration space correspond to the various physical postures of the arm as shown in Figure 4.

In an earlier experiment reported in DeMers and Kreutz-Delgado (1992), the number of solution branches $c = 4$ was assumed known. Forty thousand configuration space points were selected in a regular grid (approximately every 10^o along each joint angle), and the forward kinematic mapping was used to generate the (θ, x) data. In addition, 90 query points in one quadrant of the workspace were selected in a grid. The query points were then extended by rotating around joint axis 1.

In this paper, the experiment is repeated with 216,000 configurations selected randomly, using the data-driven Kohonen network method to choose 2000 query points. The number of inverse solutions $c = 4$ is identified correctly during the clustering phase. Figure 16 shows a projection of the points estimated to be near the singularities onto the joint angle 2–joint angle 3 plane (recall that this in fact is a torus; opposite edges are identified), along with the true singularities. Misclassification occurred for only about 5% of the configuration space.

6. Conclusions and Future Work

This paper has shown how learning methods may be used to regularize the nonlinear inverse kinematics problem for the class of admissible constant solution 3R regional manipulators described above. The approach used is based on realizing the goals of partitioning input-output data via the use of clustering and learning techniques that exploit known topological properties of the kinematic mapping. For the manipulators considered in this paper, these properties are (1) continuity; (2) the number of solutions c is constant almost everywhere over the workspace; and (3) the same c solution branches, $\mathcal{B}_i = \mathcal{A}_i$, exist almost everywhere globally over the workspace. The property of continuity allows us to effectively use the nearest neighbor clustering algorithms developed in Section 3.3. The property that c is constant almost everywhere over the workspace enables us to estimate c by applying the clustering algorithms to randomly selected data samples. The property that the same c solution branches exist almost everywhere over the workspace allows us to efficiently and unambiguously categorize the data samples by the use of algorithm LABEL.

The algorithms have been applied to admissible 3R manipulators, and it has been shown that the critical point surface manifolds in the configuration space of these manipulators can be identified, and the joint space partitioned into the posture classes (solution branches). The input-output data for each can be labeled as to posture according to the partition, and this labeled data can now be used by a supervised learning method, such as a feedforward neural network, to construct a classifier $\mathbb{B}(\theta)$ for the solution branches in configuration space.

Because the forward kinematics over each region is now one to one, it is invertible when suitably restricted. The inverse kinematics function can be approximated by function approximation methods from the position-joint angle data in order to estimate the inverse function directly. In particular, the neural network learning methods of Kuperstein (1991) or Martinetz, Ritter, and Schulten (1990) can be applied.

Thus, for Type 1 manipulators, the feasibility of constructing the branch identification function has been demonstrated, and the construction of approximate inverse kinematics directly has been shown.

Acknowledgment

This work was supported by NSF grant IRI-9202581.

References

Baker, D. R. 1990. Some topological problems in robotics. *The Mathematical Intelligencer* 12(1):66–76.

Barhen, J., Gulati, S., and Zak, M. 1989 (June). Neural learning of inverse kinematics for redundant manipulators in unstructured environments. *IEEE Computer*.

Borrel, P. 1986. Contribution a la modelisation geometrique des robots manipulateurs, application a la conception assistee par ordinateur. These d'état, USTL, Montpelier.

Burdick, J. 1988. Kinematics and design of redundant robot manipulators. Ph.D. thesis, Stanford University, Stanford, CA.

DeMers, D. E. 1993. Learning to invert many-to-one mappings. Ph.D. thesis, University of California at San Diego, La Jolla, CA.

DeMers, D. E., and Kreutz-Delago, K. K. 1992a. Learning global direct inverse kinematicss. *Advances in Neural Information Processing Systems 4*, ed. J. E. Moody et al. San Mateo, CA: Morgan Kaufmann, pp. 589–594.

DeMers, D. E., and Kreutz-Delgado, K. K. 1992b. Learning global topological properties of robot kinematic mappings for neural network based configuration control. *Neural Networks for Robotics*, ed. K. Goldberg, pp. 3–17.

DeMers, D. E., and Kreutz-Delgado, K. K. 1996. Canonical parameterization of excess motor degrees of freedom with self-organizing maps. *IEEE Trans. Neural Networks* 7(1).

Duda, R. O., and Hart, P. E. 1973. *Pattern Classification and Scene Analysis*. New York: Wiley.

Hsu, M.-S., and Kohli, D. 1987. Boundary surfaces and accessibility regions for regional structure of manipulators. *Mech. Mach. Theory* 22(3):277–289.

Jordan, M. I., and Rumelhart, D. E. 1992 (September). Forward models: Supervised learning with a distal teacher. *Cognitive Science* 16:307–354.

Kohli, D., and Hsu, M.-S. 1987. The Jacobian analysis of workspaces of mechanical manipulators. *Mech. Mach. Theory* 22(3):265–275.

Kuperstein, M. 1987 (Raleigh, NC). Adaptive visual-motor coordination in multijoint robots using parallel architecture. *Proc. 1987 Int. Conf. Robotics and Automation*, pp. 1595–1602.

Kuperstein, M. 1988. Neural model of adaptive hand-eye coordination for single postures. *Science* 239:1308–1311.

Kuperstein, M. 1991. INFANT neural controller for adaptive sensory-motor control. *Neural Networks* 4:131–145.

Martinetz, T. M., Ritter, H. J., and Schulten, K. J. 1990. Three-dimensional neural networks for learning visuomotor coordination of a robot arm. *IEEE Trans. Neural Networks* 1(1):131–136.

Pieper, D. L. 1968. The kinematics of manipulators under computer control. Ph.D. thesis, Stanford University, Stanford, CA.

Raghavan, M., and Roth, B. 1989 (Tokyo, Japan). Kinematic analysis of the 6R manipulator of general geometry. *Proc. 5th Int. Symp. Robotics Research.*

Ritter, H. J., Martinetz, T. M., and Schulten, K. J. 1989. Topology-conserving maps for learning visuo-motor-coordination. *Neural Networks* 2:159–168.

Spanos, J., and Kohli, D. 1985. Workspace analysis of regional structures. *J. Mech. Trans. Automation Design* 107:216–223.

Wenger, P. 1991. New results on the kinematics of robots: Generalizations of the aspects and classification of robots geometry. Technical Report 91.21, Laboratoire d'Automatique de Nantes, École Centrale Nantes.

Wenger, P. 1992. On the kinematics of manipulators with general geometry: Application to the feasibility analysis of continuous trajectories. *Robotics and Manufacturing*, vol. 4, ed. M. Jamshidi et al. New York: ASME Press, pp. 15–20.

Zahn, C. T. 1971. Graph-theoretical methods for detecting and describing gestalt clusters. *IEEE Trans. Comp.* 20:68–86.