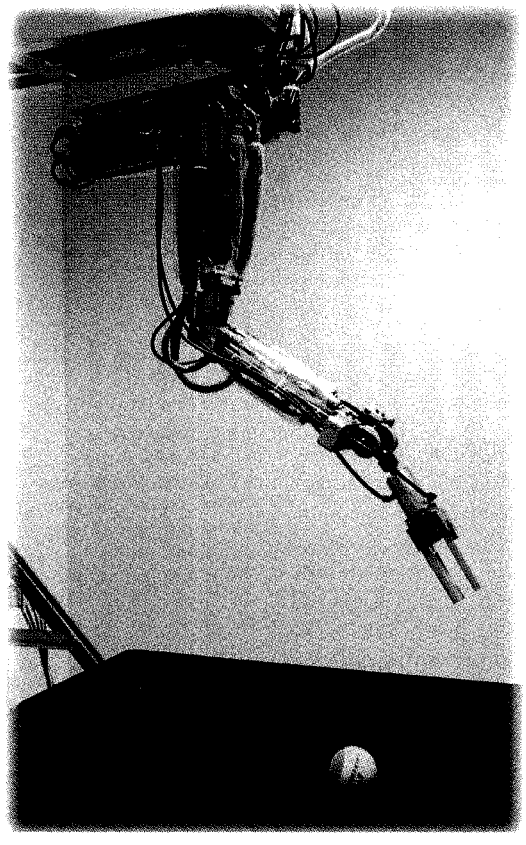
Motion Planning of a Pneumatic Robot Using a Neural Network

Michael Zeller, Rajeev Sharma, and Klaus Schulten

Integration of sensing and motion planning plays a crucial role in autonomous robot operation. We present a framework for sensor-based robot motion planning that uses learning to handle arbitrarily configured sensors and robots. The theoretical basis of this approach is the concept of the *Perceptual Control Manifold* that extends the notion of the robot configuration space to include sensor space. To overcome modeling uncertainty, the *Topology Representing Network* algorithm is employed to learn a representation of the Perceptual Control Manifold. By exploiting the topology-preserving features of the neural network, a diffusion-based path planning strategy leads to flexible obstacle avoidance. The practical feasibility of the approach is demonstrated on a pneumatically driven robot arm (*SoftArm*) using visual sensing.



Introduction

Autonomous robotics requires the generation of motion plans for achieving goals while satisfying environmental constraints. Classical motion planning is defined on a configuration space which is generally assumed to be known, implying the complete knowledge of both the robot kinematics as well as knowledge of the obstacles in the configuration space [1]. Uncertainty, however, is prevalent, which makes such motion planning techniques inadequate for practical purposes. Sensors such as cameras can help in overcoming uncertainties but require proper utilization of sensor feedback for this purpose. A robot motion plan should incorporate constraints from the sensor system as well as criteria for optimizing the sensor feedback. However, in most motion planning approaches, sensing is decoupled from planning. In [2], a framework for motion planning was proposed that considers sensors as an integral part of the definition of the motion goal. The approach is based on the concept of a *Perceptual Control Manifold* (*PCM*), defined on the product of the robot configuration space and sensor space (e.g., a set of image features). The *PCM* provides a flexible way of developing motion plans that exploit sensors effectively. However, there are robotic systems, such as the pneumatic robot arm that we use for our experiments, where the *PCM* cannot be derived analytically, since the exact mathematical relationship between configuration space, sensor space, and control signals is not known. Even if the *PCM* is known analytically, motion planning may require the tedious and error-prone process of calibration of both the kinematic and imaging parameters of the system [3,4]. Instead of using the analytical expressions

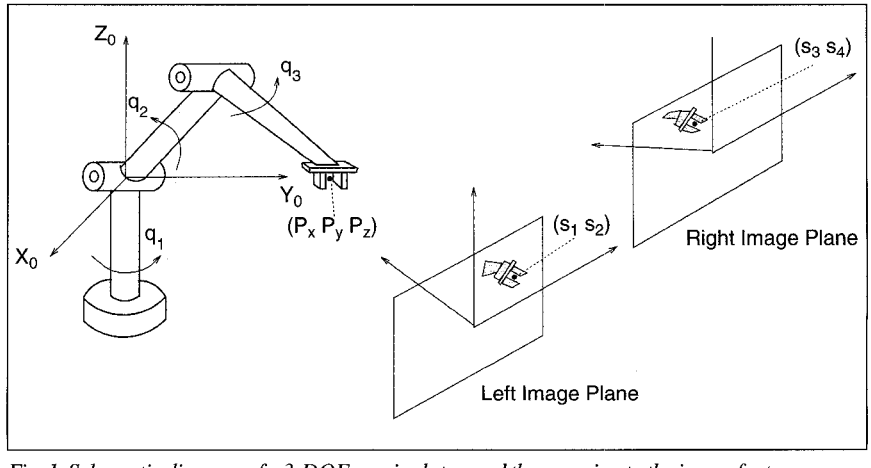


Fig. 1. Schematic diagram of a 3-DOF manipulator, and the mapping to the image feature space.

M. Zeller (zeller@ks.uiuc.edu) and K. Schulten (kschulte@ks.uiuc.edu) are with the Physics Department and the Beckman Institute for Advanced Science and Technology, University of Illinois at Urbana-Champaign. R. Sharma (rsharma@cse.psu.edu) is with the Department of Computer Science and Engineering, Pennsylvania State University.

0272-1708/97/$10.00©1997IEEE

for deriving the *PCM,* we therefore propose the use of a self-organizing neural network to learn the topology of this manifold.

In the first section, the general *PCM* concept is developed; the following section describes the *Topology Representing Network* (TRN) algorithm [5] we use to approximate the *PCM* and a diffusion-based path planning strategy which can be employed in conjunction with the TRN. The learned representation is then utilized for motion planning and control of a pneumatic robot system (*SoftArm*). Path control and flexible obstacle avoidance, as outlined in the final section, demonstrate the feasibility of this approach for motion planning in a realistic environment and illustrate the potential for further robotic applications.

Incorporating Sensor Constraints into Motion Planning

The problem of motion planning of an articulated robot is usually defined in terms of the configuration space, C (or C-space), which consists of a set of parameters corresponding to the joint variables of the robot manipulator. If q_i $(i = 1 \ldots n)$ defines each of the joint parameters and Q_i $(i = 1 \ldots n)$ defines the joint space which is the set of possible variations of each of the joint parameters, then the configuration space is defined as the n-dimensional manifold [1]

$$C \equiv Q_1 \times Q_2 \times \ldots Q_n \subseteq \Re^n .$$

The obstacles and other motion planning constraints are usually defined in terms of C, followed by the application of an optimization criteria that yields a motion plan.

In vision-based control, the robot configuration is related to a set of measurements which provide a feedback about the Cartesian position (e.g., P_x, P_y, P_z for the robot in Fig. 1) of the end-effector using the images from one or more video cameras. We assume that this feedback is defined in terms of measurable image parameters that we call *image features*, $s_i \in S_i$ $(i = 1 \ldots m)$, where S_i is the set of possible variations of each of the image features. The *image feature space* is defined as the set of all possible variations of all the m image features,

$$S \equiv S_1 \times S_2 \times \ldots S_m .$$

Before planning the vision-based motion, the set of image features must be chosen. Examples of image features used in visual servo control include image plane coordinates of a point [6-8], length, orientation, and other parameters of a line in the image [6, 9, 10], centroid, area, and other higher-order moments of an image region [10, 11], and composite features in [12]. Discussion of the issues related to feature selection for visual servo control applications can be found in [9, 10, 13]. The mapping from the set of positions and orientations of the robot tool to the corresponding image features can be computed using the projective geometry of the camera. Examples of commonly used projective geometry models include perspective, orthographic, or para-perspective projection models. Since the Cartesian position of the end-effector, in turn, can be considered to be a mapping from the configuration space of the robot, we can also define image features with a mapping from C. Thus, an image feature can also be defined as a function s_i which maps robot configurations to image feature values, $s_i : C \rightarrow S_i$. A robot trajectory in configuration space will yield a trajectory in the image feature space.

Because of the limitations of the video sensors the image features can be measured only when the objects in view are suitably configured with respect to the camera [14, 15]. Thus, to best utilize the sensor feedback, a robot motion plan should incorporate constraints from the vision system as well as criteria for optimizing the quality of the visual feedback. In [2, 16] a framework was proposed for incorporating sensor constraints in robot motion planning. The basis of this framework was the definition of the *Perceptual Control Manifold* or *PCM*. The *PCM* is a manifold defined on the product space $C \times S$, or CS-space. We know that an n-dimensional configuration space C maps to an m-dimensional feature space S. Therefore, this mapping can be defined in terms of the vector-valued function $f : C \rightarrow S$ and results in an n-dimensional manifold in an $(n + m)$-dimensional space. The *PCM* is a n-dimensional manifold since it is derived from n independent joint parameters (while the m image parameters are not independent).

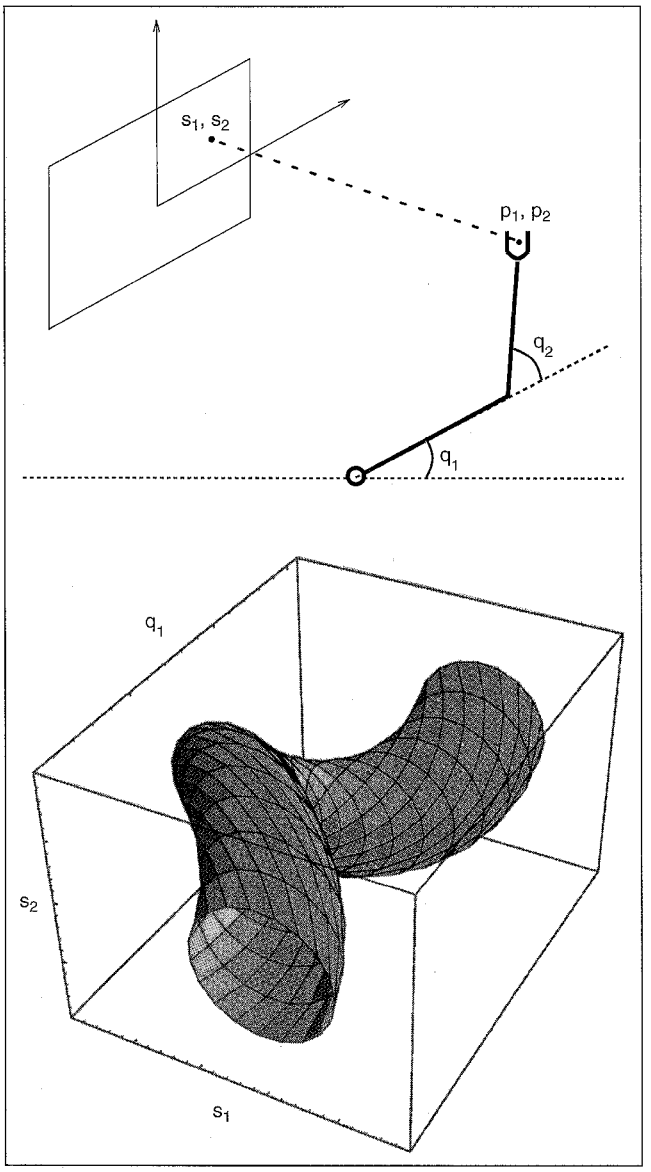


Fig. 2. A two-joint robot and the 3D projection of its PCM.

Consider a simple two-joint articulated robot shown in Fig. 2 and consider the image coordinates of the image of the end-effector s_1, s_2. The image coordinates provide a feedback on the position of the end-effector. Consider the variation of the image parameter, s_1, when the joint parameter q_1, is varied, while keeping q_2 fixed. Without considering the joint, this would define an ellipse in the Q_1 x S_1 space. Similarly, if both the joints, q_1 and q_2 are varied simultaneously, a hyper-ellipsoid will be defined in Q_1 x Q_2 x S_1 x S_2 space. For ease of visualization, we project the corresponding *PCM* to S_1 x S_2 x Q_1, as shown in Fig. 2. Analogously, in a robot with higher degrees of freedom, the *PCM* for a hand/eye setup is defined by varying all the joints and considering the parametric hypersurface defined in C x S space.

A given robot configuration maps to exactly one point on the *PCM*. The corresponding image features are not necessarily unique for a given position, but the additional representation of the joint establishes the uniqueness property needed for motion planning and control. Since the *PCM* represents both the control parameter and the sensor parameter, an appropriate control law can be defined on it [2]. Our concern in this article is, however, on motion planning. The motion planning problem can be defined in terms of the *PCM* as that of determining a trajectory to the goal position satisfying constraints presented by robot kinematics, workspace obstacles, the control system, and the visual tracking mechanism. The use of the *PCM* makes the sensor constraints easier to express compared to a potentially awkward *C*-space representation. An example of such a constraint is the avoidance of image feature singularities [17].

With a complete knowledge of the robot kinematics and camera parameters, it would be possible to model the *PCM* analytically and carry out the motion planning on this space. However, such an analytical model would be hard to derive under incomplete information, especially for a robot like the pneumatically controlled *SoftArm* that we use in our experiments. This motivates us to consider learning of the *PCM* and to subsequently use the learned space for sensor-based motion planning. In the following we introduce the neural network architecture used for learning a representation of the *PCM*.

Topology Representing Networks for Motion Planning

In order to learn a suitable mapping of the *PCM*, the neural network has to discover neighborhood relations in the input data and successfully construct a topological representation of the input manifold. Inspired by topology-conserving maps observed in many parts of the brain, several neural network algorithms have been suggested to learn such mappings [18]. A recently proposed self-organizing process is the *Topology representing network* algorithm. Topology representing networks (TRN), as introduced by Martinetz and Schulten [5], can be formulated as a combination of a vector quantization scheme and a competitive Hebb-rule. Fig. 3 illustrates the basic network architecture.

The problem of representing continuous data $\mathbf{u}$ through a discrete set $\mathbf{S} = \{\mathbf{w}_1, ..., \mathbf{w}_n\}$ of representative points is commonly referred to as vector quantization [19]. The vector quantizer implemented in the TRN is known in the literature as "neural-gas" algorithm [20]. It uses an update rule for input weights $\mathbf{w}_i^{in}$ of the network based on a ranking order and distributes the weights according to the probability density distribution of the given input data set. The adaptive neural-gas vector quantization scheme offers significant advantages compared to other classic vector quantization methods [21]. Visuo-motor control of an industrial robot using the neural-gas algorithm has been demonstrated in [22].

For the path planning task we consider here, it is also essential to capture the neighborhood relationships in the sampled data in order to model the exact topology of the *PCM*. Methods from computational geometry provide means to discover those spatial relationships in a given data structure. Therefore, TRN employs a "competitive Hebb-rule" for updating the connections between the nodes of the network based on a principle for the change of interneural connection strength which was formulated by D. Hebb [23]. According to this postulate, the connection strength between two neurons increases if their activity is correlated. In addition to correlated activity, the competitive Hebb-rule in the TRN algorithm uses closeness ranking for selecting the connections to be updated. This enforces a competition among all links c_{ij} and only the connection c_{01} between the neurons ranked 0 and 1 is updated, leading to a perfectly topology representing network structure.

To illustrate the unfolding dynamics of the map during the learning stage, Fig. 4 depicts the development of a two-dimensional network. At the beginning, all neurons are initialized with random numbers, in this case we select $\mathbf{w}_i^{in} \in [0.45, ..., 0.55]$. During the learning process the network is presented with equally distributed random numbers $\mathbf{u} = (x_1, x_2)^T, x_i \in [0 ... 1]$ and the neural-gas vector quantization scheme distributes the weights $\mathbf{w}_i^{in}$ matching the input probability distribution. Simultaneously, the competitive Hebb-rule introduces connections between the units resembling the topology of the input manifold. The quantization error q_{error} is one indication of the network quality and can be used to determine the learning time.

For the robot control task we consider later, the input weights $\mathbf{w}_i^{in}$ carry the information from joint encoders and video frames, while an additional set of output weights $\mathbf{w}_i^{out}$, indicated on the right side in Fig. 3, generates the pressure values to drive the robot. During the unsupervised training session random pressure signals are sent to the *SoftArm*. Camera and encoder readings are collected and used to train the input weights $\mathbf{w}_i^{in}$. The network acquires the topology of the *PCM* on the input side and, on the output side, it delivers the corresponding control signals to guide the robot to a desired position.

Although TRN are related to Self-Organizing Feature Maps (SOFM) [24], *a priori* knowledge of the input dimensionality is not crucial and the algorithm adjusts to the topological structure of a given input manifold $\mathbf{M}$ forming a perfectly topology preserving mapping. In many applications, the input manifold is a submanifold of a high-dimensional input space and may either be unknown or its topology may not be simple enough for prespecifying a correspondingly structured graph. For this purpose, the TRN approach is best suited because it offers a flexible way to develop a discrete representation of the underlying data structure including neighborhood relationships. For an extensive review on topology representing maps and biological brain function as well as an overview of different applications, see [25]. A rigorous definition of the terms "neighborhood preserving mapping" and "perfectly topology preserving map" based on Voronoi polyhedra and Delaunay triangulations is given in [5]. In the following, we will outline the implemented algorithm, in-

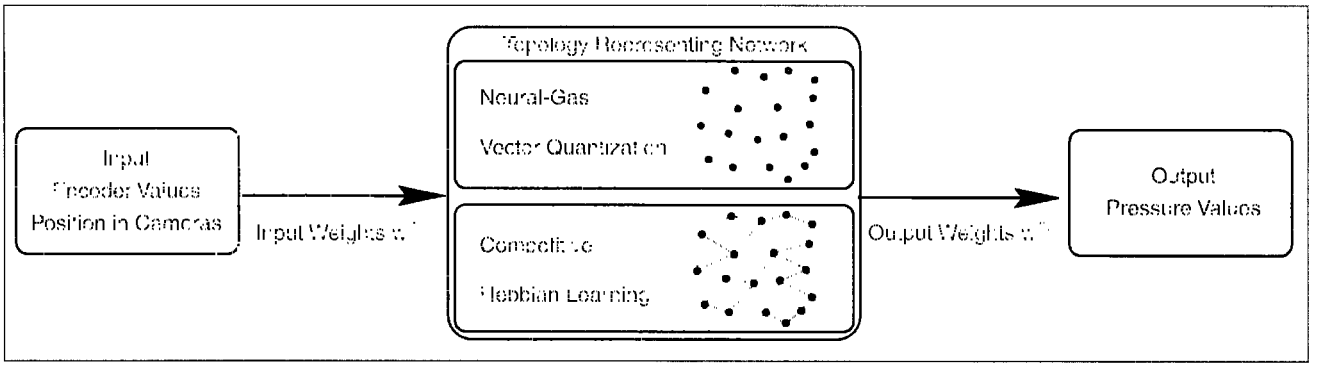


Fig. 3. Topology Representing Network: Architecture of the algorithm and input/output connections. The neural-gas vector quantization scheme distributes the pointers through adjustment of the input weights $\mathbf{w}_i^{in}$. Connections, introduced by the competitive Hebb-rule, capture the topology of the input data. Additional output weights $\mathbf{w}_i^{out}$ store the corresponding control signals for the robot.

cluding the extension of the original sequence to provide additional output weights $\mathbf{w}_i^{out}$ which will be used to link a desired control action to a specific sensory input. We assume the data to be embedded in an Euclidean space $\Re^D$ of dimension D, i.e., weights $\mathbf{w}_i \in \Re^D$ and input vector $\mathbf{u} \in \Re^D$.

Following the initialization of input weights $\mathbf{w}_i^{in}$ and output weights $\mathbf{w}_i^{out}$ for all units $i = 1 \ldots N$ with random numbers and resetting all connections to $c_{ij} = 0$ the learning cycle reads:

1. Read input vector $\mathbf{u}$ and determine the current ranking order.

$$\left\|\mathbf{w}_0^{in} - \mathbf{u}\right\| \le \left\|\mathbf{w}_1^{in} - \mathbf{u}\right\| \le \ldots \le \left\|\mathbf{w}_{N-1}^{in} - \mathbf{u}\right\| \tag{1}$$

2. Update input weights $\mathbf{w}_i^{in}$ and output weights $\mathbf{w}_i^{out}$ according to:

$$\mathbf{w}_i^{in}(t+1) = \mathbf{w}_i^{in}(t) + \gamma(r, t)\cdot\left(\mathbf{u} - \mathbf{w}_i^{in}(t)\right) \tag{2}$$

$$\mathbf{w}_i^{out}(t+1) = \mathbf{w}_i^{out}(t) + \gamma(r, t)\cdot\left(\mathbf{u} - \mathbf{w}_i^{out}(t)\right) \tag{3}$$

with

$$\gamma(r, t) = \varepsilon(t)\cdot e^{-r_i/\lambda(t)} \tag{4}$$

for $i = 1 \ldots N$, where r_i is the current rank of neuron i as determined in step 1. $\varepsilon(t)$ determines the change in the synaptic weights and $\lambda(t)$ represents a neighborhood function.

3. Update the connection c_{01} between the units currently ranked 0 and 1. If $c_{01} = 0$ then set $c_{01} = 1$ and the age of the connection $t_{01} = 0$; if $c_{01} \neq 0$ refresh the connection age, i.e., $t_{01} = 0$.

4. Increase the age of all connections c_{0j} to $t_{0j} = t_{0j} + 1$ for all units j with $c_{0j} \neq 0$. Remove connections c_{0j} which exceed a given lifetime $t_{0j} > T(t)$, i.e., have not been refreshed suitably. Continue with step 1.

Both $\varepsilon(t)$ and $\lambda(t)$ as well as $T(t)$ are a function of time and depend on the current learning step t in the same manner ($\varepsilon(t) = \varepsilon_i\left(\varepsilon_f / \varepsilon_i\right)^{t/t_{\max}}$, $\lambda(t) = \lambda_i\left(\lambda_f / \lambda_i\right)^{t/t_{\max}}$, and $T(t) = T_i\left(T_f / T_i\right)^{t/t_{\max}}$ with $\varepsilon_i = 0.3$, $\varepsilon_f = 0.05$, $\lambda_i = 0.2N$, $\lambda_f = 0.01$, $T_i = 0.1N$, $T_f = 2N$). The neural-gas vector quantization is represented by steps 1 and 2 in the algorithm above while steps 3 and 4 implement the competitive Hebb-rule which forms the topology representing manifold through the connections c_{ij}.

After the topology preserving map of the input manifold $\mathbf{M}$, which in our case is equivalent to the *PCM*, has been established, a locally optimized path can be determined by minimizing the Euclidean distance

$$d_E\left(\mathbf{w}_i^{in}, \mathbf{w}_{target}^{in}\right) = min\left\{d_E\left(\mathbf{w}_i^{in}, \mathbf{w}_{target}^{in}\right)\right\} \tag{5}$$

from the current position to a given target. Clearly, this approach will only work under very restricted conditions, e.g., a simple *PCM* manifold. In the presence of obstacles within the workspace, however, we have to use a more sophisticated procedure to develop a motion plan. We therefore propose to use a diffusion-based path finding algorithm on the discrete network lattice in which the target neuron i_t is the source of a diffusing substance [26].

The goal is to find a linked chain $i_{0,1,\ldots,n}$ on the graph leading from the current position $i_0 = i_c$ to the target position $i_n = i_t$, while satisfying certain constraints, e.g., obstacles in the workspace of the robot arm (see Fig. 5).

To find the desired path, we define a function $f_i(t)$ on the nodes i of the network obeying the condition:

$$f_t(t) = 1 \quad \forall t \tag{6}$$

and the relaxation dynamics:

$$f_i(t+1) = \frac{m}{N_i}\sum_{j\in F_i} f_j(t) \quad \text{if } i \neq i_t \tag{7}$$

The function $f_i(t)$, initially set to $f_i(0) = 0$ $(i \neq i_t)$, represents the concentration at each node i of the network and is held constant at the target node i_t while diffusing through the links of the network. F_i denotes the set of all nodes which are neighbors of i as defined by the network topology and N_i is the number of nodes in F_i. A flux can be defined as the concentration difference between two connected nodes and is directed toward the node with lower concentration. A value $m < 1$ corresponds to absorptive losses at the nodes. In order to avoid the trivial stationary solution we choose

$$m = \frac{N_i}{N_i + 1} \tag{8}$$

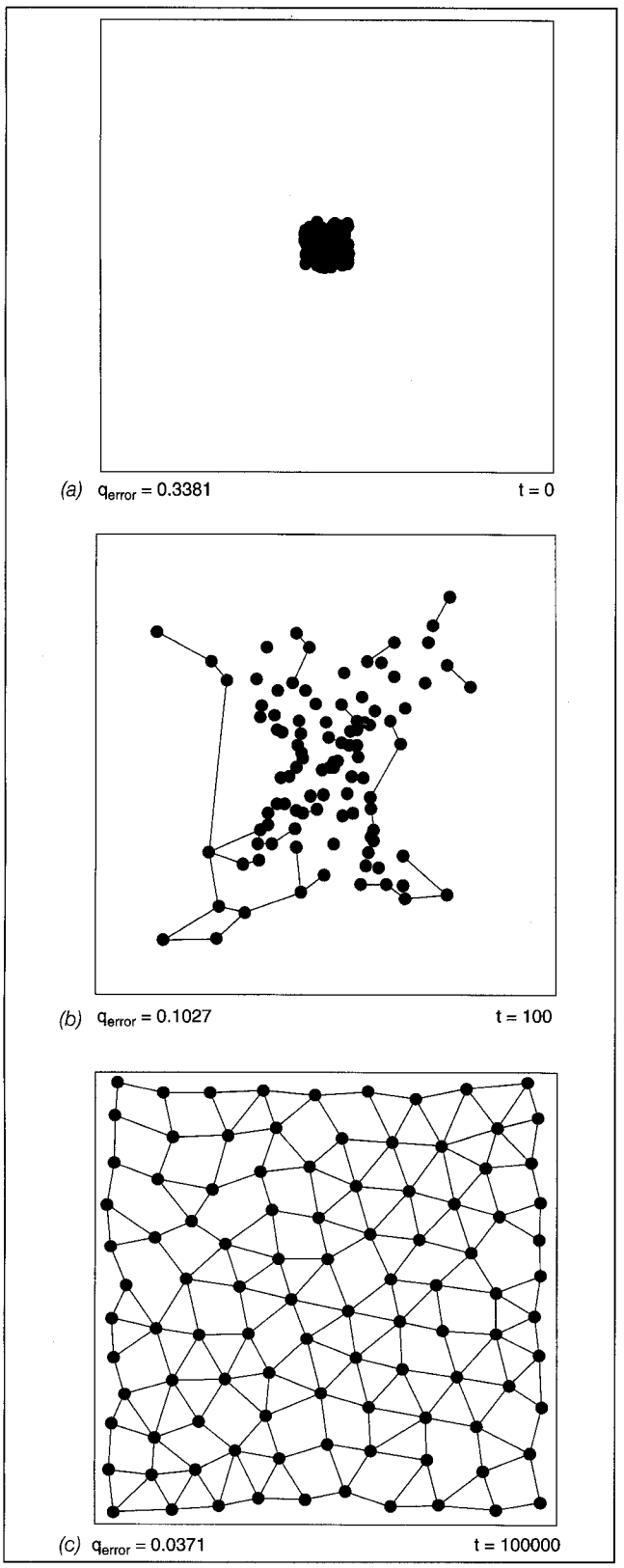


Fig. 4. Topology representing network: Learning process, showing the development of the mapping. Left: Initial distribution of neurons. Middle: An intermediate stage after the first connections have been established. Right: The final network resembles the topology of the feature space.

While other relaxation procedures can be used as well, this simple relaxation scheme serves our purpose and is computationally inexpensive, an important step toward real-time control, e.g., in the presence of moving obstacles.

The path leading from i_c to i_t can be found by starting at the current node i_c and choosing as the next step always the neighbor with maximal $f_i(t)$. Since the network graph is finite, the algorithm is guaranteed to terminate yielding a proper path $i_{c,1,2,\ldots,n\text{-}1,t}$. The path is short in the sense that it takes a route that maximizes the increase of $f_i(t)$ at each step.

To summarize, the motion plan can be generated as follows:

1. Read current position $\mathbf{u}_c$ and target position $\mathbf{u}_t$ in visual space.
2. Find best matching neurons as given by the input weights $\mathbf{w}_c^{in}$ and $\mathbf{w}_t^{in}$
3. Detect obstacles in vision space and eliminate connections to neurons i which are covered by an obstacle.
4. Define diffusion on network lattice and iterate until $f_i \neq 0$

$$f_i(t+1) = \begin{cases} 1 & \text{if } i = i_t \\ \dfrac{1}{N_i + 1} \displaystyle\sum_{j \in F_i} f_j(t) & \text{else} \end{cases} \tag{9}$$

5. Follow steepest gradient of $f_i(t)$ from current unit i_c to target unit i_t

In step 3 one can include other constraints, for example avoiding singularities in the sensor space [2]. It is not necessary to iterate step 4 until a stationary solution of the diffusion has been achieved. Instead, we can generate a path as soon as the concentration on the current node i_c is larger than zero. This might, however, render a recomputation of the diffusion necessary, if, for any reason, the current location is displaced into a position that has not been covered by the process yet. Finally, if the motion plan meets a given goal, movement can be initiated using the corresponding output values $\mathbf{w}_i^{out}$ of the map to generate the sequence of commands, e.g., to guide a robot arm from start to target.

Other graph search algorithms and global optimization strategies can be applied to the learned representation of *PCM* as well. However, these will be computationally expensive especially when complex obstacles are taken into account [1]. Our approach on the other hand is of complexity $O(N^2)$ with N being the number of nodes in the network and, in particular, it is independent of the number and shape of obstacles. Furthermore, the plan is resolution complete; only if the resolution of the discretized *PCM* manifold is not high enough to resolve a possible path, the algorithm will fail to find it. Following the steepest gradient of $f_i(t)$ from i_c to i_t eliminates the problem of local minima associated with many potential field path planning methods [1].

In Fig. 6 we plot a sample path on a two-dimensional network (N = 500) from the upper left unit to the upper right using the diffusion algorithm. Instead of learning a static topology including obstacles, we initially present the complete workspace during the training stage and dynamically map obstacles into the *PCM* after the representation of the workspace has been accomplished. This approach is more suited for a robotic manipulator operating in a changing environment, e.g., with obstacles placed at different locations within the workspace.

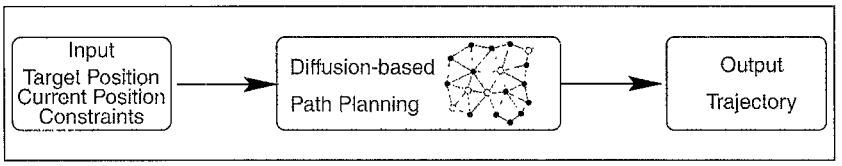


Fig. 5. Path planning: Given the current and the target position of the robotic manipulator as well as certain constraints, e.g., obstacles in the workspace, the diffusion algorithm generates a trajectory on the TRN.

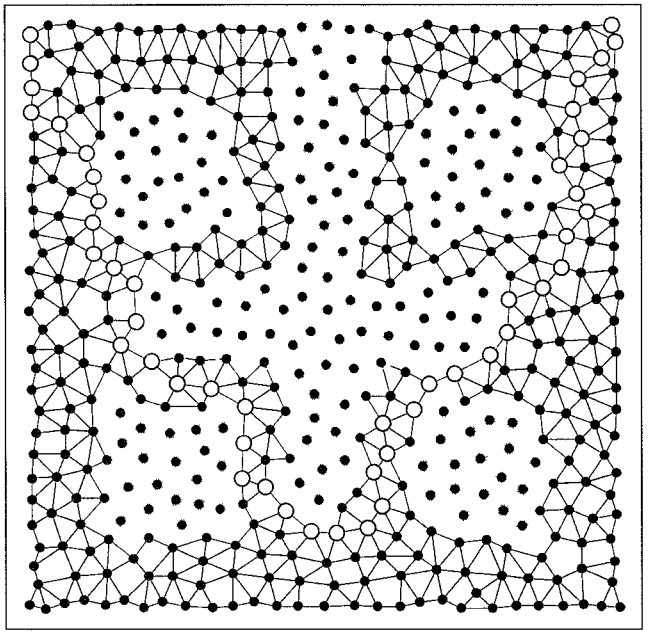


Fig. 6. Two-dimensional Topology Representing Network after the learning has been finished. Obstacles have been identified and a sample path (open circles) from upper left unit to upper right unit of the map has been generated by the diffusion process.

As a means of demonstrating the practical capabilities for motion planning and control, the following sections will describe the *SoftArm* robotic system and the implementation of the topology representing network algorithm on this system.

The *SoftArm* Robotic System

The *SoftArm*, a pneumatically driven robotic manipulator manufactured by Bridgestone, is modeled after the human arm. It exhibits the essential mechanical characteristics of skeletal muscle systems employing agonist-antagonist pairs of *rubbertuators* which are mounted on opposite sides of rotating joints. Pressure difference drives the joints; average pressure controls the force (compliance) with which the motion is executed. This latter feature allows operation at low average pressures and, thereby, allows one to carry out a compliant motion of the arm. This makes such robots suitable for operation in a fragile environment; in particular, it allows direct contact with human operators. The price to be paid for this design is that the response of the arm to pressure signals $(\bar{p}_1, \bar{p}_2, \ldots, \bar{p}_N)^T$ cannot be described by *a priori* mathematical equations, but rather must be acquired heuristically. Furthermore, one expects that the response characteristics change during the lifetime of the arm through wear, after replacement of parts and, in particular, through hysteretic effects. In consequence, accurate positioning of the *SoftArm* presents a challenging problem and can only be achieved by an adaptive control mechanism. For a more detailed introduction to the mechanics of the *SoftArm* see [27].

The complete robot system, which is depicted in Fig. 7, consists of the *SoftArm*, air supply, control electronics (servo drive units) and a Hewlett Packard HP755/99 workstation which includes a serial interface connected to the robot's servo drive units, and a video input card (Parallax Power Video 700Plus). The servo drive units provide the internal control circuitry of the robot, operate the servo valve units, and send joint angle data, available from optical encoders mounted on each joint, to the computer. Visual feedback is provided by two color video cameras. For maximum flexibility, vision processing is implemented in software rather than in hardware. The use of a frame grabber to import the video signals in a JPEG encoded format minimizes the amount of data to be transferred between the video board and workstation memory. The location of the gripper is extracted from the video frames through a simple color separation, yielding one color component. This is then thresholded and the center of mass of the remaining image calculated. Coding the gripper in a certain color, e.g., red, allows us to weaken the workspace scenery restrictions in terms of background and lighting conditions while, at the same time, keeping the visual preprocessing as simple and efficient as possible.

Motion Planning for the *SoftArm* Robotic System

Previous work on topology representing networks (TRN) in robotics [27, 28] employed neighborhood preservation to average over the output of several adjacent units in order to speed up learning and to achieve a more accurate positioning. The present study seeks to exploit the topology to generate a motion plan from a current position to a given target satisfying several constraints. These constraints can include obstacles defined in C-space, obstacles given through vision space and limitations of the camera feedback [17].

The *PCM*, as introduced above, is defined as the product of *C*-space and sensor space *S*. Therefore, two different types of information converge upon neurons within the network. Visual input $\mathbf{s} = (s_1 \ldots s_n)^T$ is derived from video cameras; vision preprocessing resolves the gripper location in the video frames. Joint position of the manipulator, denoted by $\mathbf{q} = (q_1 \ldots q_n)^T$, is derived from the feedback of optical encoders mounted on each joint. Following a suitable training period, the topology of the neural network resembles the *PCM*. In addition, the network provides the nonlinear mapping between the position in work space $\mathbf{u} = (\mathbf{s}, \mathbf{q})^T$ and the corresponding pressure commands $\mathbf{p}$ to achieve this configuration. The current experiments focus on obstacle collisions of the robot's gripper only. Future studies will extend the control to avoid obstacles with the complete arm. Given the exact robot geometry, this can be implemented by using the encoder information which is already stored at each network node.

First we test our approach in a two-dimensional environment, generating a motion plan in one camera plane. Therefore, we use only two joints of the *SoftArm* to control the position. In this case, the network provides a mapping between the 4d input vector $\mathbf{u}$ and the two-dimensional pressure vector $\mathbf{p}$:

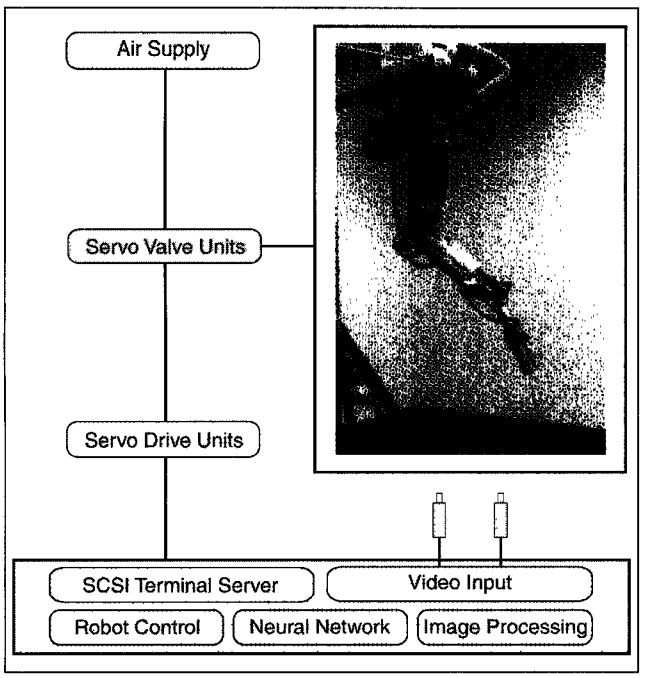


Fig. 7. Diagram of the robot system, showing SoftArm, air supply, control electronics, and workstation. The host computer includes a software layer (robot control, neural network, and image processing programs) and the hardware components (serial interface and video input).

$$input: \mathbf{u} = (s_1, s_2, q_1, q_2)^T \tag{10}$$

$$output: \mathbf{p} = (p_1, p_2)^T \tag{11}$$

A sample network is depicted in Fig. 8 by plotting the visual components **s** of the 4-dimensional input vectors $\mathbf{w}_i^{in}$. This network was trained ($t_{max} = 100000$) with a data set of 2700 random moves within a subset of the workspace and consists of $N = 150$ neural units. The left side shows the actual position in the robot's workspace as seen by the camera. On the right side, we use the learned representation to generate a motion plan from a start point to a given target. Both, start and target, are only given in visual space **s** (as is the obstacle); the corresponding encoder readings need not to be known. By selecting the best matching neurons for current position and target position in vision space, the resulting neurons also provide the values for the encoder readings. This is possible, because **s** and **q** represent redundant information. The motion plan, shown in Fig. 8 on the right hand side, is generated by the diffusion algorithm exclusively in $\mathcal{CS}$-space to ensure a smooth motion in terms of joint angles.

Extending the algorithm to a three-dimensional workspace increases the information that needs to be processed by the network. The image feature space $\mathcal{S}$ is now represented by the position $\mathbf{s} = (s_1, s_2, s_3, s_4)^T$ of the gripper in two camera planes, the configuration space $\mathcal{C}$ is given by the encoder readings **q** of three joints, resulting in a 7d input vector and a three-dimensional output vector for the pressure signals respectively:

$$input: \mathbf{u} = (s_1, s_2, s_3, s_4, q_1, q_2, q_3)^T \tag{12}$$

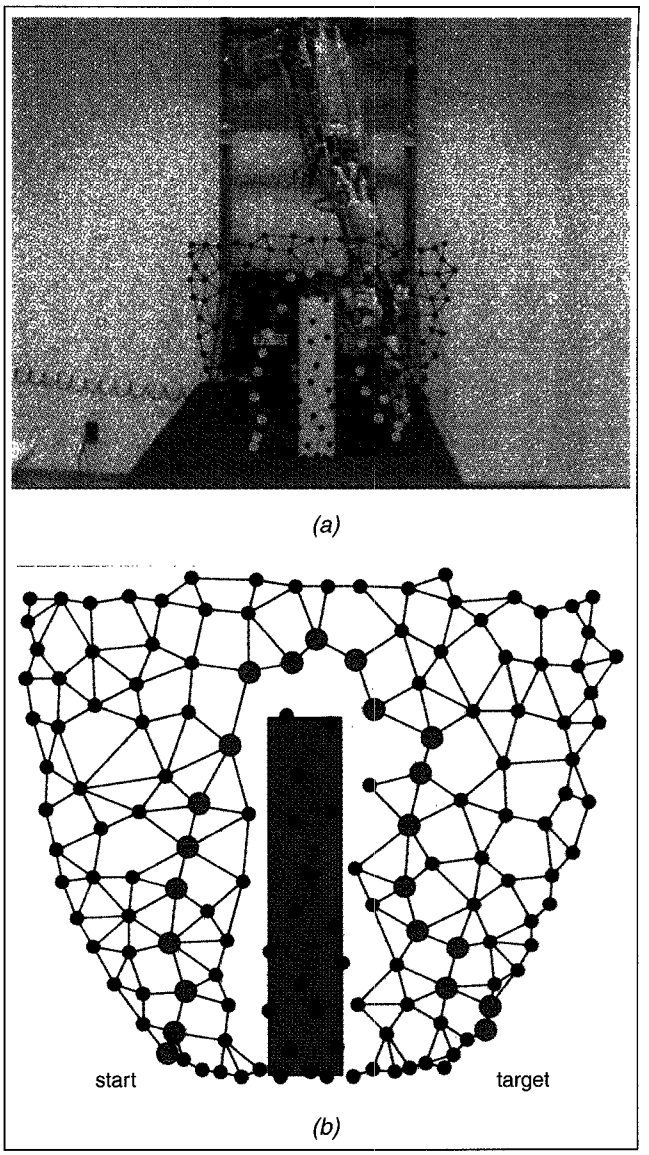


Fig. 8. Left: SoftArm robot system, obstacle and network structure in the workspace as seen by the camera. The learning has been accomplished and the network represents the topology of the PCM. Right: Motion plan (grey units) generated by the diffusion process in $\mathcal{CS}$-space after start and target have been defined in vision space.

$$output: \mathbf{p} = (p_1, p_2, p_3)^T \tag{13}$$

In this case, a network consisting of $N = 750$ neurons is trained ($t_{max} = 250000$) with 5000 random moves within the workspace by sending random pressure values to the robot and observing the end effector position as well as reading out the encoder values.

The sequence in Fig. 9 shows a sample path. The robot arm is moving from the right side to the left side of the workspace while avoiding a collision with the obstacle placed in the middle. The training of the neural network was restricted to the lower portion of the workspace, so that no valid path over the obstacle exists in this case. Again, start and target location as well as obstacles are

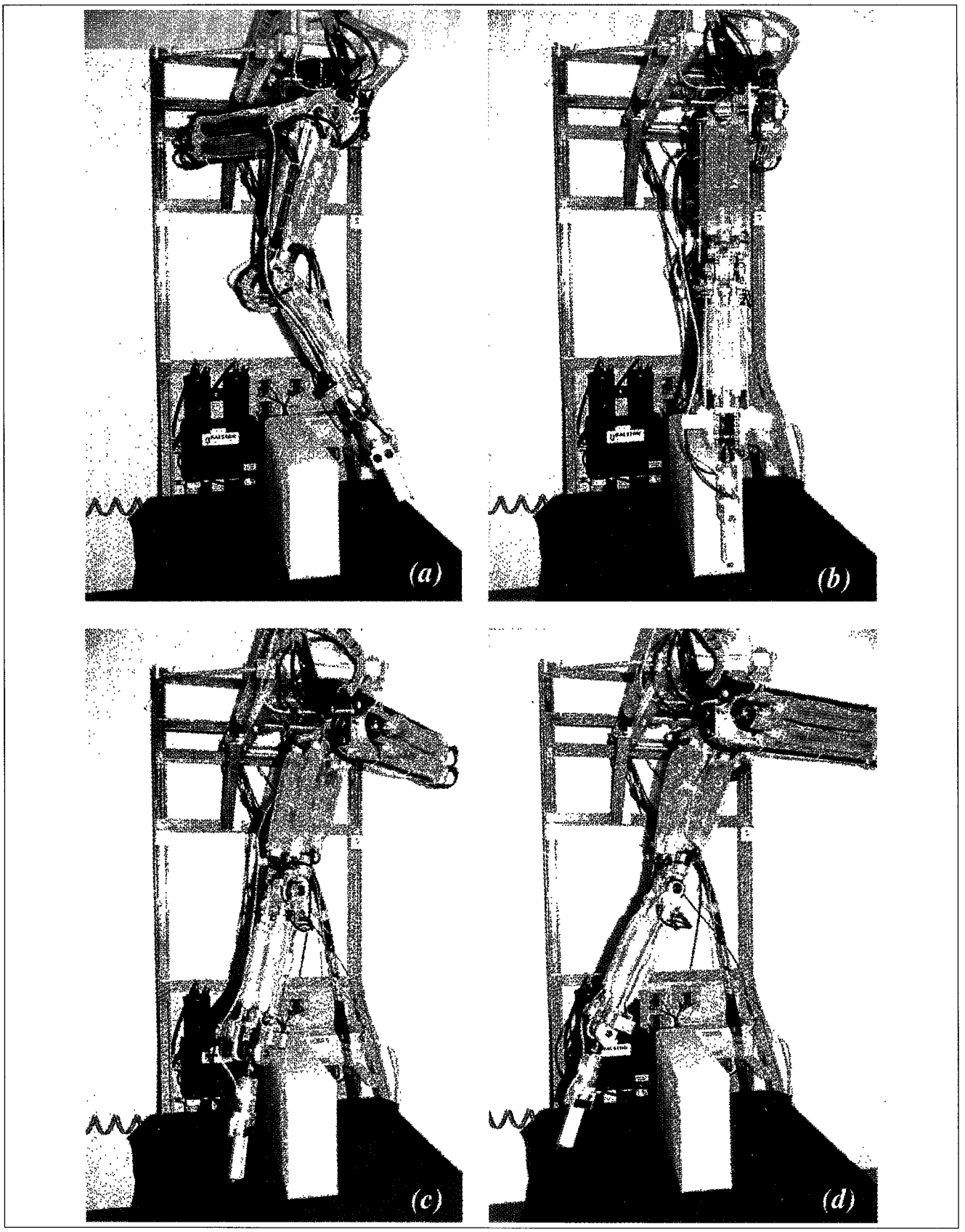


Fig. 9. 3d path generated by the diffusion process. The successive frames show the robot arm moving from an initial configuration on the right side of the workspace to a given target position on the left side while avoiding a collision with the detected obstacle.

only known in vision space, while the motion plan is generated on the learned representation of the *PCM* by the diffusion procedure. In the three-dimensional environment, a problem arises due to the discrete representation of the *PCM*. With our network of 750 neurons, the resulting path generated on the learned representation of the *PCM* is only 12 steps long which, in a more realistic environment, can only be seen as a rough motion plan. Nevertheless, it can be taken as a piecewise linear approximation of the final path and a basis for further path smoothing techniques [1]. Neural network interpolation strategies can be used to improve the accuracy as well [27].

While collecting the large data set of random robot arm moves for training the neural network takes several hours, the TRN can typically be trained within a few minutes on a workstation. Calculating the diffusion path plan takes less than one second on an HP 755/99 workstation for the experiments we mention here.

Discussion

Learning the representation of the *Perceptual Control Manifold* provides a very general framework for robot motion planning in which the sensing (in the form of video feedback) is factored automatically into the planning process, leading to a flexible way of visually controlling a robot manipulator. The implementation on a pneumatically driven robot manipulator proves the technical feasibility of our method. It can be generalized to control robotic systems with more degrees of freedom in a three-dimensional environment as our experiments demonstrate. The proposed diffusion-based path planning algorithm utilizes the topology preserving features of the neural network to dynamically map obstacles into the *PCM* and to establish a motion plan which prevents collisions with detected obstacles. Not taking into account hardware-specific limitations such as camera and encoder resolution, the accuracy of a motion plan is only limited by the network size. A larger number of neurons leads to a finer sampling of the underlying *PCM* topology and, hence, to a more detailed path, essential for an environment with complex obstacles. In contrast to industrial robot systems, the *SoftArm* has not been designed to facilitate accurate posture control. A simulator environment is not available and the mechanics of the arm confine the size of our data base to a few thousand moves, leading to a coarse representation of the manifold. Hence, the number of neurons in our experiments is lower than it would be in a typical implementation using an industrial robot system. Future work, however, will have to address the discretization effect in higher dimensions, as introduced by the use of redundant degrees of freedom, to achieve a finer path control. This discretizing effect that results from the use of small numbers of neurons to map a high dimensional input space can be alleviated by introducing interpolation strategies [27] which also improve fine motion control.

Acknowledgments

The authors would like to thank Ivo Hofacker and Willy Wriggers for critical comments on an earlier version of this manuscript. This work was supported by the Roy J. Carver Charitable Trust and by the U.S. Army Research Laboratory under Cooperative Agreement No. DAAL01-96-2-0003. Simulations were carried out at the Resource for Concurrent Biological Computing at the University of Illinois, funded by the National Institute of Health (Grant P41RR05969) and by the National Science Foundation (Grants BIR-9318159 and ASC-8902829).

References

[1] J.C. Latombe, *Robot Motion Planning*, Kluwer Academic, Boston, MA, 1991.

[2] R. Sharma and H. Sutanto, "A Framework for Robot Motion Planning with Sensor Constraints," *IEEE Transactions on Robotics and Automation*, vol. 13, no. 1, pp. 61-73, 1997.

[3] R.Y. Tsai, "Synopsis of Recent Progress on Camera Calibration for 3D Machine Vision," in *Robotics Review 1*, MIT Press, Cambridge, MA, 1989.

[4] J.M. Hollerbach, "A Survey of Kinematic Calibration," in *Robotics Review 1*, O. Khatib, J.J. Craig, and T. Lozano-Perez, eds., MIT Press, Cambridge, MA, 1989.

[5] T. Martinetz and K. Schulten, "Topology Representing Networks," *Neural Networks*, vol. 7, no. 3, pp. 507-522, 1994.

[6] B. Espiau, F. Chaumette, and P. Rives, "A New Approach to Visual Servoing in Robotics," *IEEE Transactions on Robotics and Automation*, vol. 8, pp. 313-326, 1992.

[7] N.P. Papanikolopoulos, P.K. Khosla, and T. Kanade, "Visual Tracking of a Moving Target by a Camera Mounted on a Robot: A Combination of Vision and Control," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 1, pp. 14-35, 1993.

[8] S.B. Skaar, W.H. Brockman, and R. Hanson, "Camera-Space Manipulation," *International Journal of Robotics Research*, vol. 6, no. 4, pp. 20-32, 1987.

[9] J.T. Feddema, C.S. George Lee, and O.R. Mitchell, "Weighted Selection of Image Features for Resolved Rate Visual Feedback Control," *IEEE Transactions on Robotics and Automation*, vol. 7, pp. 31-47, 1991.

[10] L.E. Weiss, A.C. Sanderson, and C.P. Neuman, "Dynamic Sensor-Based Control of Robots with Visual Feedback," *IEEE Transactions on Robotics and Automation*, vol. 3, pp. 404-417, 1987.

[11] B. Yoshimi and P.K. Allen, "Active, Uncalibrated Visual Servoing," in *Proc. IEEE International Conference on Robotics and Automation*, San Diego, CA, May 1994, pp. 156-161.

[12] G.D. Hager, "Real-Time Feature Tracking and Projective Invariance as a Basis for Hand-Eye Coordination," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1994, pp. 533-539.

[13] H. Sutanto and R. Sharma, "Global Perfomance Evaluation of Image Features for Visual Servo Control," *Journal of Robotic Systems*, vol. 13, no. 4, pp. 243-258, April 1996.

[14] R. Sharma, "Active Vision for Visual Servoing: A Review," in *IEEE Workshop on Visual Servoing: Achievements, Applications and Open Problems*, May 1994.

[15] K.A. Tarabanis, P.K. Allen, and R.Y. Tsai, "A Survey of Sensor Planning in Computer vision," *IEEE Transactions on Robotics and Automation*, vol. 11, pp. 86-104, 1995.

[16] R. Sharma and H. Sutanto, "Unifying Configuration Space and Sensor Space for Vision-Based Motion Planning," in *Proc. IEEE International Conference on Robotics and Automation*, April 1996, pp. 3572-3577.

[17] R. Sharma and S. Hutchinson, "Optimizing Hand/Eye Configuration for Visual-Servo Systems," in *Proc. IEEE International Conference on Robotics and Automation*, May 1995, pp. 172-177.

[18] Helge Ritter, Thomas Martinetz, and Klaus Schulten, *Neural Computation and Self-Organizing Maps: An Introduction*, Addison-Wesley, New York, revised English edition, 1992.

[19] R.M. Gray, "Vector Quantization," *IEEE ASSP Magazine*, vol. 1(2), pp. 4-29, 1984.

[20] T. Martinetz and K. Schulten, "A 'Neural Gas' Network Learns Topologies," in *Proceedings of the International Conference on Artificial Neural Networks, Helsinki, 1991*. 1991, Elsevier Amsterdam.

[21] T. Martinetz, S.G. Berkovich, and K. Schulten, "Neural-Gas Network for Vector Quantization and Its Application to Time-Series Prediction," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 558-569, 1993.

[22] J.A. Walter and K. Schulten, "Implementation of Self-Organizing Neural Networks for Visuo-Motor Control of an Industrial Robot," *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 86-95, 1993.

[23] D. Hebb, *Organization of Behavior*, Wiley, New York, 1949.

[24] T. Kohonen, "Analysis of a Simple Self-Organizing Process," *Biol. Cybern.*, vol. 44, pp. 135-140, 1982.

[25] Klaus Schulten and Michael Zeller, "Topology Representing Maps and Brain Function," in *Nova Acta Leopoldina NF 72*. 1996, vol. 294, pp. 133-157, Deutsche Akademie der Naturforscher.

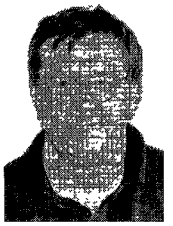
[26] H. Ritter and K. Schulten, "Planning a Dynamic Trajectory Via Path Finding in Discretized Phase Space," in *Parallel Processing: Logic, Organization, and Technology*, vol. 253 of *Lecture Notes in Computer Science*, pp. 29-39. Springer, 1987.

[27] T. Hesselroth, K. Sarkar, P. van der Smagt, and K. Schulten, "Neural Network Control of a Pneumatic Robot Arm," *IEEE Transactions of System, Man and Cybernetics*, vol. 24, no. 1, pp. 28-37, 1994.

[28] K. Sarkar and K. Schulten, "Topology Representing Network in Robotics," in *Physics of Neural Networks, Volume 3*, J. Leo van Hemmen, Eytan Domany, and Klaus Schulten, eds. Springer-Verlag, New York, 1995.

Michael Zeller received his Diplom degree in physics from the Johann Wolfgang Goethe University in Frankfurt, Germany, in 1994. He is currently a visiting scholar in the Physics Department and Beckman Institute at the University of Illinois at Urbana-Champaign. His research interests include vision-based path planning for robotic manipulators, biological visuo-motor control, neural networks and nonlinear dynamics.

Rajeev Sharma received his Ph.D. from the University of Maryland, College Park, in 1993. He is currently an Assistant Professor in the Department of Computer Science and Engineering at the Pennsylvania State University, University Park. He spent three years at the University of Illinois, Urbana-Champaign, as a Beckman Fellow and Adjunct Assistant Professor in the Department of Electrical and Computer Engineering. He is a recipient of the ACM Samuel Alexander Doctoral Dissertation Award, IBM pre-doctoral fellowship, National Talent Search Scholarship of India, and National Merit Scholarship Award. He is a co-editor of the electronic newsletter of the IEEE assembly and task planning subcommittee. His research interests include motion planning under uncertainty, vision-based control, neural networks, active vision, and vision-based human-computer interaction.



Klaus Schulten received his Diplom degree in physics from the University of Munster, Germany, in 1969, and the Ph.D. degree in chemical physics from Harvard University in 1974. In 1974 he joined the Max-Planck-Institute for Biophysical Chemistry in Gottingen and in 1980 he became Professor of Theoretical Physics at the Technical University of Munich. In 1988 he moved to the University of Illinois at Urbana-Champaign, where he is Professor of Physics. His research areas are theoretical physics and computational biology.