Vision-Based Motion Planning of a Pneumatic Robot Using a Topology Representing Neural Network

Michael Zeller

Rajeev Sharma

Klaus Schulten

Beckman Institute for Advanced Science and Technology
University of Illinois at Urbana-Champaign
405 N. Mathews Avenue
Urbana, IL 61801

email: zeller@ks.uiuc.edu - rajeev@cs.uiuc.edu

Abstract

We present a framework for sensor-based motion planning of robotic manipulators using a *topology representing network* (TRN). Exploiting the perfectly topology preserving features of the network, the algorithm learns the representation of the *Perceptual Control Manifold* (PCM), a recently introduced concept for motion planning. This concept allows sensors to be integrated into robot motion planning. Besides a demonstration of the technical feasibility of motion planning through perfectly topology preserving maps the capabilities of this approach within an engineering framework, namely the implementation on a pneumatically driven robot arm (*SoftArm*), are demonstrated.

Keywords: *motion planning, topology representing neural network.*

Submitted to IEEE 1996 Int. Symposium on Intelligent Control

1 Introduction

An important step toward autonomous robotics is developing ways to generate motion plans for achieving certain goals while satisfying environmental constraints. Classical motion planning is defined on a configuration space ($\mathcal{C}$-space) which is assumed to be known, implying the complete knowledge of both the robot kinematics as well as knowledge of the obstacles in the $\mathcal{C}$-space [7]. Uncertainty, however, is prevalent which makes some of these motion planning techniques quite inadequate for practical purposes. Sensors such as cameras can help in overcoming these uncertainties. To best utilize the sensor feedback, a robot motion plan should incorporate constraints from the sensor system as well as criteria for optimizing the quality of the sensor feedback. Unfortunately, in most motion planning approaches, sensing is completely decoupled from planning. In [15] we present a framework for motion planning that considers sensors as an integral part of the definition of the motion goal. The approach is based on the concept of *Perceptual Control Manifold* (*PCM*), defined on the product of the robot $\mathcal{C}$-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 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 (see section 3). Instead of using the analytical expressions for deriving the *PCM* we therefore propose the use of a self-organizing neural network to learn the topology of this manifold. Section 3 describes the *topology representing network* algorithm [9] we use to approximate the *PCM* . The learnt representation can then be utilized for motion planning and control of the robot. Experiments on a pneumatic robot system establish the feasibility of this framework for motion planning within an engineering environment. Path control is demonstrated in 2d as well as in 3d.

2 Incorporating Sensor Constraints into Motion Planning

Sensors such as the video camera have limited range of operation and work well only when the objects in view are optimally configured with respect to the camera [13, 18]. Thus, to best utilize the sensor feedback, a robot motion plan should incorporate constraints from the sensor system as well as criteria for optimizing the quality of the sensor feedback. The role of vision in a motion planning framework, such as the one based on configuration space [7], is to give an estimate of the current position of the robot in the configuration space with respect to the desired goal. For handling uncertainty in sensing of the robot position different motion planning approaches have been suggested (e.g., [2, 1]). However, there is no general mechanism for including the variation of sensing parameters and sensor constraints into the motion plan. There is thus a growing need for extending the configuration space planning paradigm to bridge the gap between planning and sensing so that the motion plans can benefit by optimally utilizing the available sensing mechanism. In [15] we proposed a motion planning framework that achieves this with the help of a space called the *Perceptual*

Control Manifold or *PCM* . The *PCM* is a manifold defined on the product of the robot configuration space (defined in terms of the set of robot joint parameters) and image feature space (defined in terms of a set of image features from the image of the robot hand). These concepts are formalized next.

Configuration space $\mathcal{C}$: The problem of motion planning of an articulated robot is usually defined in terms of the configuration space, $\mathcal{C}$ (or $\mathcal{C}$-space), which consists of a set of parameters corresponding to the joint variables of the robot manipulator. $\mathcal{C}$ is an n-dimensional manifold [7] for an n-DOF robot manipulator, i.e., $\mathcal{C} \equiv \mathcal{Q}_1 \times \mathcal{Q}_2 \times \ldots \mathcal{Q}_n \subseteq \mathbf{R}^n$, when $q_i \in \mathcal{Q}_i$ is a joint parameter (See Figure 1). The obstacles and other motion planning constraints are usually defined in terms of $\mathcal{C}$, followed by the application of an optimization criteria that yields a motion plan.

Image feature space $\mathcal{S}$: In vision-based control, the robot configuration is related to a set of measurements which provide a feedback about the cartesian position 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

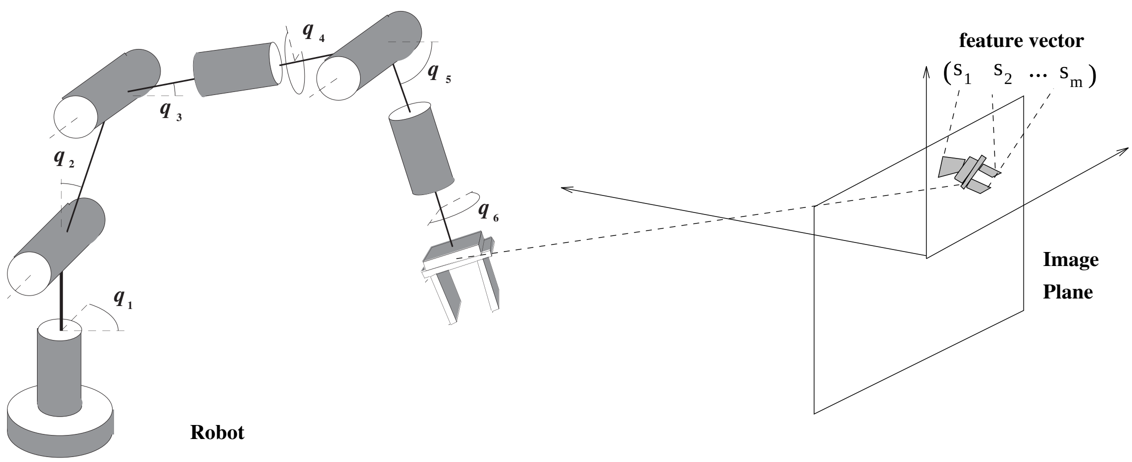

Figure 1: Schematic diagram of a 6-DOF manipulator, and the mapping to the image feature space.

we call *image features*, s_i (See Figure 1). Before planning the vision-based motion, a set of m image features must be chosen. Discussion of the issues related to feature selection for visual servo control applications can be found in [3, 17, 20]. 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 $\mathcal{C}$. Thus, an image feature can be defined as a function s_i which maps robot configurations to image feature values, $s_i : \mathcal{C} \rightarrow \mathcal{S}_i$. The set of all possible variations of the image features is termed *image feature space*, $\mathcal{S} \equiv \mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \mathcal{S}_m$.

A robot trajectory in configuration space will yield a trajectory in the image feature space.

Although we refer to vision, the discussion applies to any other sensor as well, and the term "image" is thus used for the generic sensor measurements. Moreover, we refer to *geometric* image features (e.g., position, size, distance, or surface area of objects in the image), as opposed to *photometric* features (pixel intensities, colors, etc.). Examples of image features used in visual servo control include image plane coordinates of a point, length, orientation and other parameters of a line in the image, centroid, area, and other higher order moments of an image region.

In the following, we consider a hand/eye setup where the image features are derived from stationary cameras. Other examples where a *fixed-eye* configuration is used are [6, 16]. For visual servoing examples in which an *eye-in-hand* setup is used, see [10, 20], and for examples in which an *active* camera is used, see [13, 14, 21].

Perceptual Control Manifold *PCM* **:** In order to include the image feature space, $\mathcal{S}$, in the planning space, we consider the $\mathcal{C} \times \mathcal{S}$ space, or $\mathcal{CS}$-space. We know that an n-dimensional configuration space $\mathcal{C}$ maps to an m-dimensional feature space $\mathcal{S}$. This mapping can be defined in terms of the vector-valued function $f : \mathcal{C} \rightarrow \mathcal{S}$. This mapping defines an n-dimensional manifold in $(n + m)$-dimensional space. We call this manifold the *perceptual control manifold*, or (*PCM*) and use it for developing a motion planning framework.

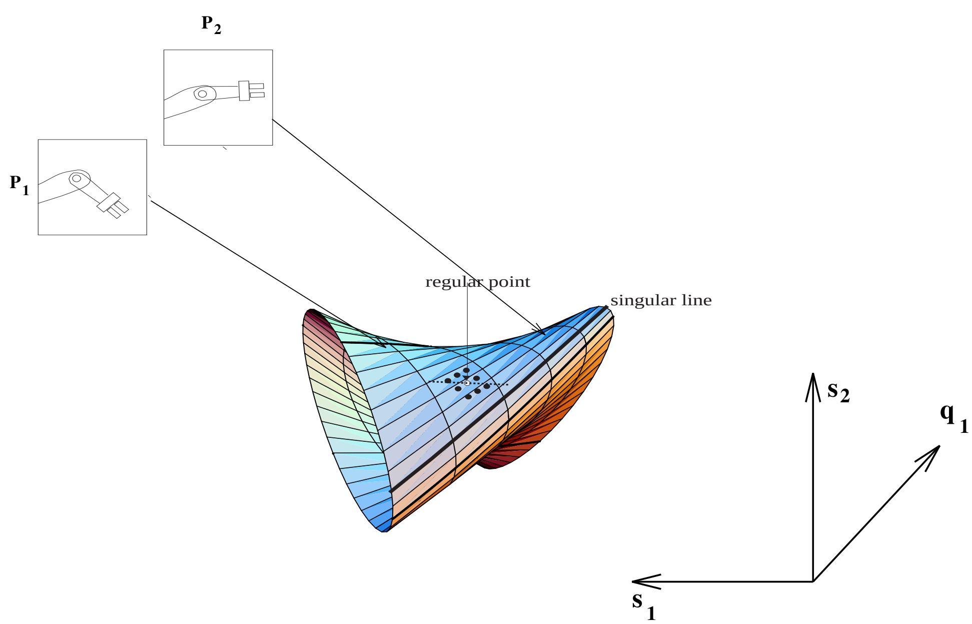

Figure 2: Positions of the manipulator mapped into the *PCM*.

For the robot in Figure 1, consider the variation of an image parameter, s_1, when a joint parameter, say q_1, is varied, while keeping the rest of the joints fixed. Without considering the joint limits for the time being, this would define an ellipse in the $\mathcal{Q}_1 \times \mathcal{S}_1$ space. Similarly, when two of the joints, say q_1 and q_2 are varied simultaneously, a hyper-ellipsoid will be defined in $\mathcal{Q}_1 \times \mathcal{Q}_2 \times \mathcal{S}_1 \times \mathcal{S}_2 \subseteq \mathbf{R}^4$. For ease of visualization, we project the corresponding *PCM* to $\mathcal{S}_1 \times \mathcal{S}_2 \times \mathcal{Q}_1 \subseteq \mathbf{R}^3$, as shown in Figure 2. Analogously, in higher dimensions, the *PCM* for a hand/eye setup is defined by varying all the joints and considering the parametric hypersurface defined in $\mathcal{Q} \times \mathcal{S}$ 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 because the joint is also represented this leads to the uniqueness property that is 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 [15]. Our concern in this paper is, however, on motion planning. An important characterization of the *PCM* for motion planning is the distinction between the *singular* and *regular* portions of the surface. The singularities of the *PCM* need to be identified so that they may be avoided during manipulation. The singularity constraints can be incorporated into the motion planning along with other constraints that are defined on the *PCM* .

A robot task task can be defined as a problem of trajectory planning on the *PCM* from the initial position of the manipulator to some goal position on *PCM*. This motion planning requires the system to satisfy constraints presented by robot kinematics, the control system and the visual tracking mechanism. The aim is to get a feasible solution space for motion planning that satisfies these constraints. In current approaches to motion planning, different types of constraints are handled independently by each subsystem. This makes it hard to achieve optimal performance, because there is no unified framework to consider all type of constraints simultaneously; the solution obtained by considering one type of constraint at a time may not yield a globally optimal solution. Further, the use of the *PCM* makes some of the sensor constraints easier to express compared to a potentially awkward C-space representation. An example, of such a constraint is image feature singularity [14]. Constraints can be classified as being *hard* or *soft*. Hard constraints, e.g., joint velocity limits, must be satisfied in a motion plan. Soft constraints, e.g., image singularity avoidance, should preferably be satisfied, and can be included with the help of a cost function in motion planning.

Optimizing motion plans: Once the necessary hard constraints have been applied to yield a feasible solution space defined on the *PCM*, any path on the *PCM*, from the point corresponding to the initial position of the robot to a desired point on *PCM*, will give rise to a valid solution. However, the path chosen should be the one that optimizes a set of desired objectives. Some of these objectives could be related purely to the robot, e.g., minimizing joint movement, minimizing relative velocity of the robot end-effector and a moving target, etc. ; others could be related purely to the sensor, e.g., maximizing the variation in the image features. However, there are some objectives that involve both the robot and the sensor. These objectives directly effect the control and mainly depend on how the image features vary with an infinitesimal movement of the robot. The objective could be to steer away from image singularities or to find the robot trajectory such that it follows a singularity. The *PCM* framework has the advantage that it allows a variety of optimization criteria to be expressed in a unified manner so that the optimal sensor-based plan can be generated for interception.

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, as mentioned in Section 1, such an analytical model would be hard to derive under incomplete information, especially for a robot like the pneumatically controlled arm that we use in our experiments. This motivates us to consider learning of the *PCM* and using the learned space for sensor-based motion planning. We next consider the neural network architecture used for learning a representation of the *PCM* .

3 Topology Representing Networks for Motion Planning

Topology representing networks (TRN), as introduced by Martinetz and Schulten [8, 9], can be formulated as a combination of a vector quantization scheme and a competitive Hebb rule. Although related to Self-Organizing Feature Maps (SOFM) [5], *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. A rigorous definition of the terms 'neighborhood preserving mapping' and 'perfectly topology preserving map' based on Voronoi polyhedra and Delaunay triangulations is given in [9]. In the following, we will outline the implemented algorithm, including 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.

Following the initialization of input weights $\mathbf{w}_i^{in}$, 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 current ranking order.

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

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

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

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

   with

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

   for $i = 1 \ldots N$, where r_i is the current rank of neuron i as determined in step 1. $\epsilon(t)$ 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 $c_{01} > 0$ refresh the connection age.

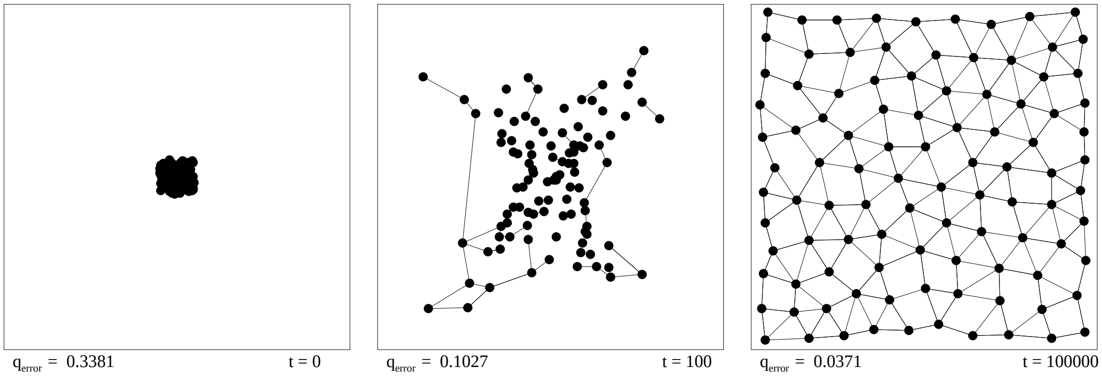

Figure 3: *Topology representing network:* Learning process, showing the development of the mapping. (*left*) initial distribution of neurons; (*middle*)an intermediate stage after the first connections have been established; (*right*) the final network resembles the topology of the feature space.

4. Increase the age of all connections c_{0j} to $t_{0j} = t_{0j} + 1$ for all units j with $c_{0j} > 0$. Remove connections c_{0j} which exceed a given lifetime $t_{0j} > T(t)$. Continue with step 1.

Both $\epsilon(t)$ 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[1].

Figure 3 shows the development of a 2d network. At the beginning, all neurons are initialized with random numbers, in this case $\mathbf{w}_i^{in} \in [0.45, \ldots, 0.55]$. During the learning process the network is presented with equally distributed random numbers $\mathbf{u} = (x_1, x_2)^T, x_i \in [0 \ldots 1]$ and the vector quantization scheme matches the input probability distribution. The competitive Hebb rule introduces connections between the units which resemble 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.

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 from the current position to a given target. The motion plan can be generated as follows:

1. Read current position $\mathbf{u}_{current}$ and target position $\mathbf{u}_{target}$
2. Find best matching neurons $\mathbf{w}_{current}^{in}$ and $\mathbf{w}_{target}^{in}$
3. Move from current unit $\mathbf{w}_{current}^{in}$ to a neighboring unit i with $c_{current,i} > 0$ that satisfies

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

[1] $\epsilon(t) = \epsilon_i(\epsilon_f/\epsilon_i)^{t/t_{max}}$ $\quad \lambda(t) = \lambda_i(\lambda_f/\lambda_i)^{t/t_{max}}$ $\quad T(t) = T_i(T_f/T_i)^{t/t_{max}}$
with $\epsilon_i = 0.3$, $\epsilon_f = 0.05$, $\lambda_i = 0.2N$, $\lambda_f = 0.01$, $T_i = 0.1N$, $T_f = 2N$

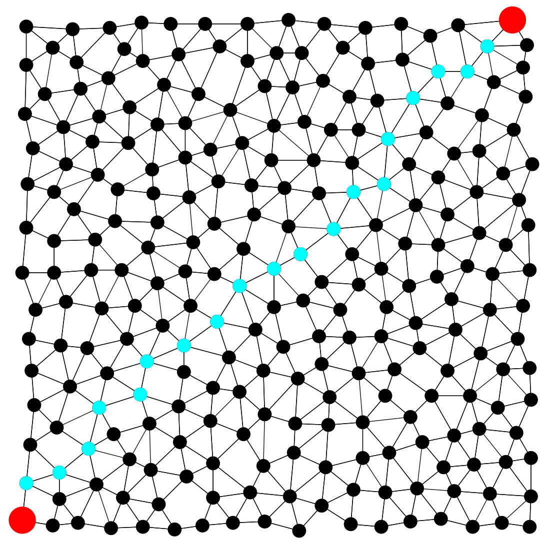

Figure 4: *2d topology representing network* after the learning has been finished. A sample path from lower left unit to upper right unit of the map has been generated by locally minimizing the Euclidean distance between current and target position.

4. If $\mathbf{w}^{in}_{current} = \mathbf{w}^{in}_{target}$ then stop, otherwise continue with step 1.

In Figure 4 we plot a sample path on a 2d network from the lower left unit to the upper right using this algorithm.

In the presence of obstacles within the workspace, step 3 has to check if a move will result in a collision and avoid it. Finally, if the motion plan meets a given goal, movement can be initiated using the corresponding output values $\mathbf{w}^{out}_i$ of the map to generate the sequence of commands necessary to navigate the robot from start to target. Other global optimization strategies can be applied to the learnt representation of *PCM* as well. However, these will be computationally expensive especially when complex obstacles are taken into account. A promising algorithm, based on a diffusion process, which we plan to explore in this regard is described in [11].

As means of demonstrating the practical capabilities within an engineering framework 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 pneumatic robot arm.

4 The *SoftArm* Robotic System

The *SoftArm* is a pneumatically driven robotic manipulator, 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, allows direct contact with human operators. The price to be paid for this design is that the response of the arm to pressure signals $(\bar{p}_1, \bar{p}_2, \ldots, \bar{p}_N)^T$ and $(\Delta p_1, \Delta p_2, \ldots, \Delta p_N)^T$ cannot be described by *a priori* mathematical equations, but rater must be acquired heuristically. Furthermore, one expects that the response characteristics change during the life time 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 [4].

The complete robot system, which is depicted in Figure 5, consists of the *SoftArm*, air supply, control electronics (servo drive units) and an 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.

5 Motion Planning for the *SoftArm* Robotic System

In addition to the previous work on topology representing networks (TRN) in robotics [19, 4, 12], where neighborhood preservation has been used to average over the output of several adjacent units in order to achieve a more accurate positioning, in the present study we focus on exploiting the topology to generate a motion plan from a current position to a given target in a 2-dimensional plane satisfying several constraints. These contstraints can include obstacles defined in C-space, obstacles given through vision space and limitations of the camera feedback [14].

The *PCM* , as introduced in Section 2, is defined as the product of $\mathcal{C}$-space and sensor space $\mathcal{S}$. Therefore, two different types of information converge upon neurons within the network. Visual input $\mathbf{s} = (s_1 \ldots s_n)^T$ is

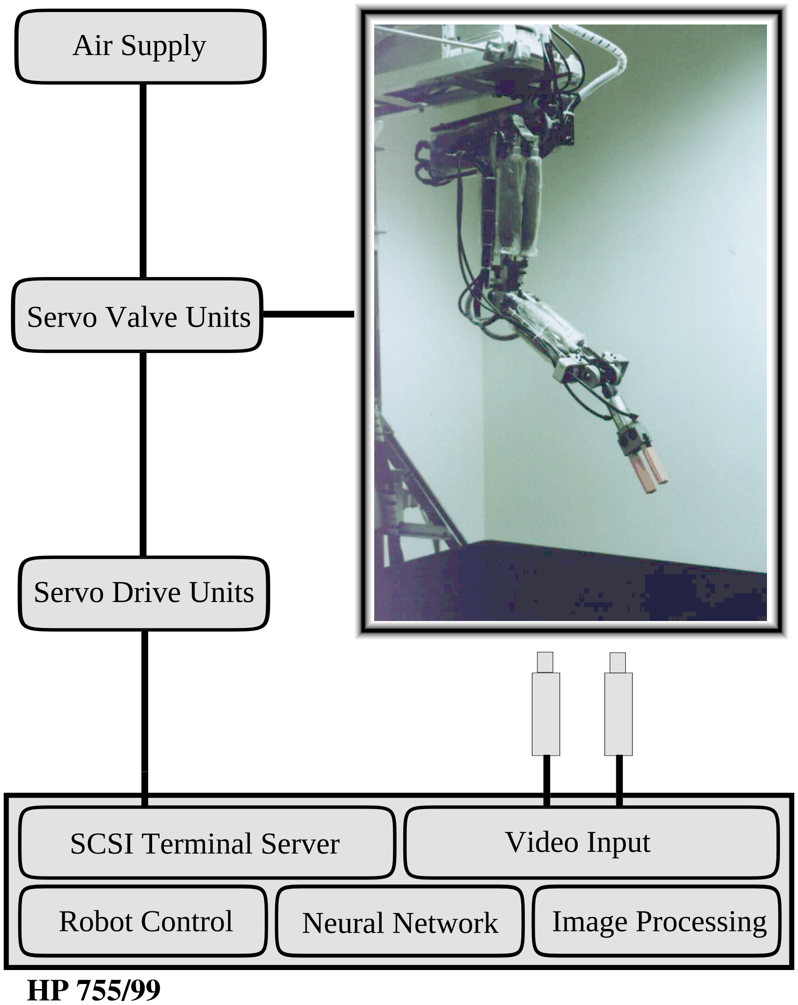

Figure 5: Diagram of the robot system, showing *SoftArm*, air supply, control electronics and workstation. The host computer includes a software layer (robot control, neural network and image processing programs) and the hardware components (serial interface and video input).

derived from video cameras; vision preprocessing resolves the gripper location in the video frames. Angular position of the manipulator, denoted by $\mathbf{q} = (q_1 \dots q_n)^T$, is derived from the feedback of optical encoders mounted on each joint. Following a suitable training period, the topology of the 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.

2d control: First we test our approach in a 2d environment, generating a motion plan in one camera plane. Therefore, we use 2 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 2d pressure vector $\mathbf{p}$.

$$\begin{aligned} \mathit{input}: \quad \mathbf{u} &= (s_1, s_2, q_1, q_2)^T && (6) \\ \mathit{output}: \quad \mathbf{p} &= (p_1, p_2)^T && (7) \end{aligned}$$

A sample network is depicted in Figure 6 by plotting the visual components $\mathbf{s}$ of the 4-dimensional input vectors $\mathbf{w}_i^{in}$. This network was trained with a dataset of 800 random moves within a subset of the workspace and consists of 75 neural units. The left side shows the actual position in the robot's workspace. On the right side, we use the learnt representation to generate a motion plan from a start point to a given target. Both, start and target, are only given in visual space $\mathbf{s}$ (as would be obstacles), 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 $\mathbf{s}$ and $\mathbf{q}$ represent redundant information. The motion plan, shown in Figure 6 on the right hand side, finally is generated exclusively in C-space to ensure a smooth motion in terms of joint angles.

3d control: Extending the algorithm to a 3d 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 2 camera planes, the configuration space $\mathcal{C}$ is given by the encoder readings $\mathbf{q}$ of 3 joints, resulting in a 7d feature vector and a 3d output vector respectively.

$$\begin{aligned} \mathit{input}: \quad \mathbf{u} &= (s_1, s_2, s_3, s_4, q_1, q_2, q_3)^T && (8) \\ \mathit{output}: \quad \mathbf{p} &= (p_1, p_2, p_3)^T && (9) \end{aligned}$$

In this case, the network is trained with 1000 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. In Figure 7 we plot a sample path in the camera planes and the corresponding encoder readings. Start and target are given in vision space, while the motion plan is generated on the learnt representation of the *PCM* by the procedure mentioned in Section 3. As the main problem in the 3d environment, we can identify the discretizing effect. With a network of 750 neurons, the resulting path generated on the learnt representation of the *PCM* is only 12 steps long, which in a realistic environment can only be seen as a

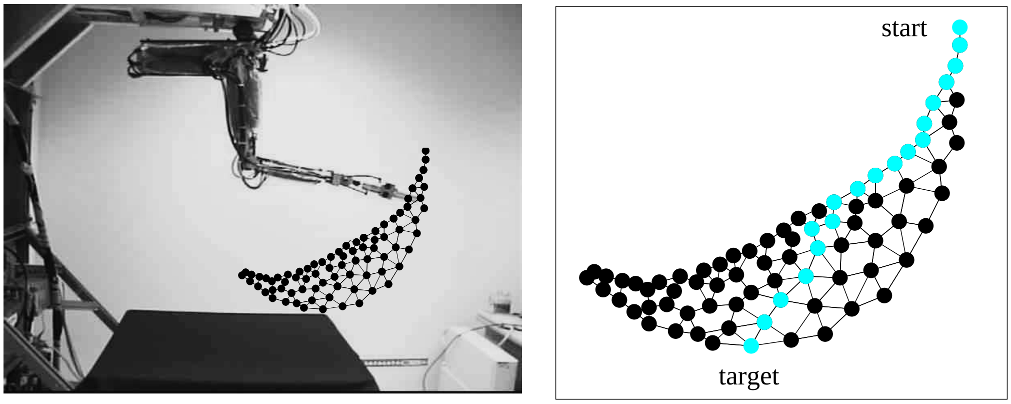

Figure 6: (*left*) *SoftArm* robot system and network structure in the workspace as seen by the camera. The learning has been accomplished and the network represents the topology of the *PCM* . (*right*) Visual components of the mapping and a motion plan (grey units) generated in configuration (encoder) space after start and target have been defined in vision space.

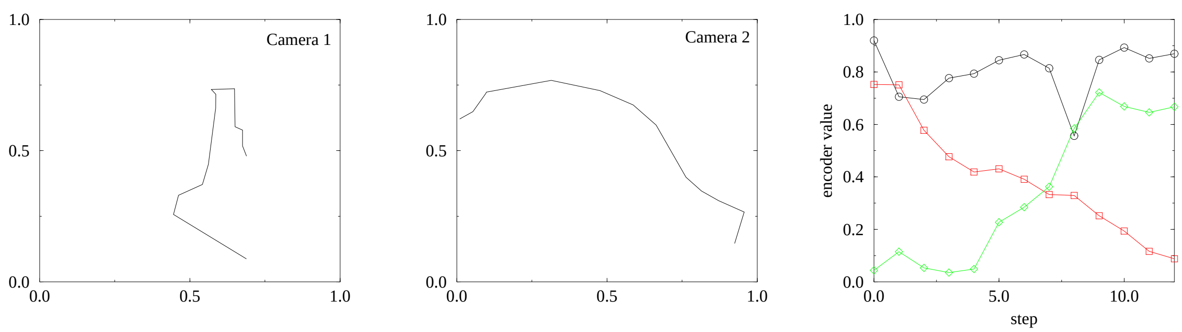

Figure 7: *3d path* in vision and encoder space: The image features $(s_1, s_2, s_3, s_4)^T$ are given by the position in camera plane 1 and 2, the encoder readings $(q_1, q_2, q_3)^T$ represent the configuration space $\mathcal{C}$.

rough motion plan. Nevertheless, it can be taken as a piecewise linear approximation of the final path. 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 [4] which also improve fine motion control.

Instead of learning a static topology including obstacles, one may initially present the complete workspace during the training stage and 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.

6 Conclusions

Learning the representation of the *perceptual control manifold* (*PCM*) 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 2d 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 3d environment as our experiments demonstrate. Future work, however, will have to address the discretisation effect in higher dimensions to achive fine motion control on the *SoftArm*. Furthermore, we are planning to implement more sophisticated path planning strategies on basis of the learnt representation of the *PCM* and study the role of redundancy for planning.

Acknowledgements: This work was supported by the Carver Charitable Trust.

References

[1] B. R. Donald and J. Jennings. Sensor interpretation and task-directed planning using perceptual equivalence classes. In *Proc. IEEE International Conference on Robotics and Automation*, pages 190–197, Sacramento, CA, April 1991.

[2] M. Erdmann. Using backprojections for fine motion planning with uncertainty. *International Journal of Robotics Research*, 5(1):19–45, 1986.

[3] 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*, 7:31–47, 1991.

[4] 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*, 24(1):28–37, 1994.

[5] T. Kohonen. Analysis of a simple self-organizing process. *Biol. Cybern.*, 44:135–140, 1982.

[6] A. J. Koivo and N. Houshangi. Real-time vision feedback for servoing robotic manipulator with self-tuning controller. *IEEE Transactions on Systems, Man, and Cybernetics*, 21:134–142, 1991.

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

[8] T. Martinetz and K. Schulten. A 'neural gas' network learns topologies. In *Proceedings of the International Conference on Artificial Neural Networks, Helsinki, 1991*. Elsevier Amsterdam, 1991.

[9] T. Martinetz and K. Schulten. Topology representing networks. *Neural Networks*, 7(3):507–522, 1994.

[10] 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*, 9(1):14–35, 1993.

[11] H. Ritter and K. Schulten. Planning a dynamic trajectory via path finding in discretized phase space. In *Parallel Processing: Logic, Organization, and Technology*, volume 253 of *Lecture Notes in Computer Science*, pages 29–39. Springer, 1987.

[12] K. Sarkar and K. Schulten. Topology representing network in robotics. In J. Leo van Hemmen, Eytan Domany, and Klaus Schulten, editors, *Physics of Neural Networks, Volume 3*. Springer–Verlag, New York, 1995.

[13] R. Sharma. Active vision for visual servoing: A review. In *IEEE Workshop on Visual Servoing: Achievements, Applications and Open Problems*, May 1994.

[14] R. Sharma and S. Hutchinson. Optimizing hand/eye configuration for visual-servo systems. In *Proc. IEEE International Conference on Robotics and Automation*, pages 172–177, May 1995.

[15] R. Sharma and H. Sutanto. Integrating configuration space and sensor space for vision-based robot motion planning. *IEEE Transactions on Robotics and Automation*, 1996. (to appear).

[16] S. B. Skaar, W. H. Brockman, and R. Hanson. Camera-space manipulation. *International Journal of Robotics Research*, 6(4):20–32, 1987.

[17] H. Sutanto and R. Sharma. Global perfomance evaluation of image features for visual servo control. *Journal of Robotic Systems*, 1996. (to appear).

[18] K. A. Tarabanis, P. K. Allen, and R. Y. Tsai. A survey of sensor planning in computer vision. *IEEE Transactions on Robotics and Automation*, 11:86–104, 1995.

[19] 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*, 4(1):86–95, 1993.

[20] L. E. Weiss, A. C. Sanderson, and C. P. Neuman. Dynamic sensor-based control of robots with visual feedback. *IEEE Journal of Robotics and Automation*, 3:404–417, 1987.

[21] J. Y. Zheng, Q. Chen, and A. Tsuji. Active camera guided manipulation. In *Proc. IEEE International Conference on Robotics and Automation*, pages 632–638, 1991.