Learning the Perceptual Control Manifold for Sensor-Based Robot Path Planning

M. Zeller, K. Schulten

Beckman Institute
and Department of Physics
University of Illinois at Urbana-Champaign
405 N. Mathews Avenue
Urbana, IL 61801
{zeller,kschulte}@ks.uiuc.edu

R. Sharma

Department of
Computer Science and Engineering
Pennsylvania State University
317 Pond Laboratory
University Park, PA 16802-6106
rsharma@cse.psu.edu

Abstract

The Perceptual Control Manifold *is a recently introduced concept that extends the notion of the robot configuration space to include sensor feedback for robot motion planning. In this paper, we propose a framework for sensor-based robot motion planning using the* Topology Representing Network *algorithm to develop a learned representation of the* Perceptual Control Manifold. *The topology preserving features of the neural network lend themselves to yield, after learning, a diffusion-based path planning strategy for flexible obstacle avoidance. Simulations on path control and flexible obstacle avoidance demonstrate the feasibility of this approach for motion planning and illustrate the potential for further robotic applications.*

1 Introduction

Autonomous robotics requires the ability to generate motion plans that achieve specified goals while satisfying environmental constraints. Classical motion planning is generally defined in terms of a configuration space or $\mathcal{C}$-space [2]. In most motion planning approaches, the $\mathcal{C}$-space is assumed to be known, implying a complete knowledge of both robot kinematics and obstacles. To address this issue, a framework for motion planning that considers sensors as an integral part of the definition of the motion goal was proposed in Refs. [7]. The approach is based on the concept of a *Perceptual Control Manifold* (*PCM*), defined on the product of the robot $\mathcal{C}$-space and sensor space. The *PCM* provides a flexible way of developing motion plans that exploit sensors effectively. The configuration space framework can be extended to include sensor space constraints such as visibility, motion perceptibility and sensor singularity. The expected result of *PCM*-based motion planning is a path which is not only collision-free but also more efficient to control, i.e., optimizes the sensor feedback.

In many practical robotic systems the *PCM* cannot be derived analytically, since the exact mathematical relationship between configuration space, sensor space and control signals is not known [12]. 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 [9]. Instead of using the analytical expressions for deriving the *PCM* we propose, therefore, the use of a self-organizing neural network to learn the topology of this manifold. Once the *PCM* is learned, it can be used as a basis for sensor-based motion planning and control.

2 Perceptual Control Manifold

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 [2] for an n-degrees of freedom robot manipulator, i.e., $\mathcal{C} \equiv \mathcal{Q}_1 \times \mathcal{Q}_2 \times \ldots \mathcal{Q}_n \subseteq \mathbf{R}^n$, where $q_i \in \mathcal{Q}_i$ 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 criterium 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 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*,

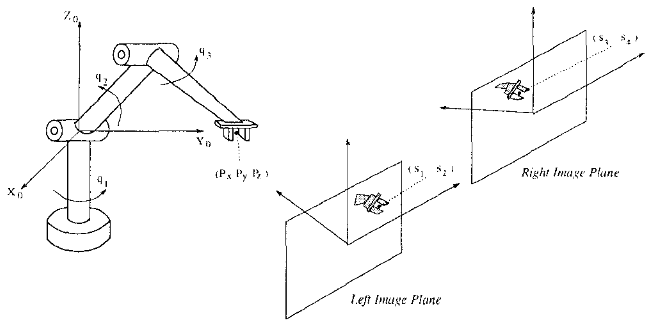

Figure 1: Schematic diagram of a 3-degree of freedom manipulator, and the mapping to the image feature space.

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 Refs. [1, 8, 11]. 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. 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.

The *Perceptual Control Manifold* or *PCM* is defined as the manifold defined on the product space $\mathcal{C} \times \mathcal{S}$, or $\mathcal{CS}$-space. We know that an n-dimensional configuration space $\mathcal{C}$ maps to an m-dimensional feature space $\mathcal{S}$. Therefore, this mapping can be defined in terms of the vector-valued function $f : \mathcal{C} \rightarrow \mathcal{S}$ and results in an n-dimensional manifold embedded in an $(n+m)$-dimensional 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 [7].

A robot task can be defined as a problem of trajectory planning on the *PCM* from the initial position of the manipulator to some goal position. This motion planning requires the system to satisfy constraints presented by robot kinematics, 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 are image feature singularities [6].

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 used for our experiments in [12]. For an industrial robot like the PUMA, described in Section 4, the kinematic model can be determined, but it is tedious to calibrate the camera setup especially if the cameras are frequently moved. This motivates us to acquire the *PCM* through a learning procedure and to subsequently use the learned space for sensor-based motion planning.

3 Topology Representing Networks for Motion Planning

The topology representing network [4] (TRN) algorithm which we will briefly outline in the following offers a flexible way to develop a discrete representation of a data structure including neighborhood relationships. TRN is a combination of *neural gas* and *competitive Hebbian learning*. While the neural gas [3] part, a soft-competitive learning algorithm, distributes the input weights according to the input probability distribution, a competitive Hebb-rule preserves the topology by introducing neighborhood relations using a winner-take-all principle. Knowledge of the input dimensionality is not crucial and the algorithm automatically adjusts to the topology of the input manifold. Even when 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, the TRN will construct a perfectly topology preserving mapping. Therefore, the TRN algorithm presents a good choice for learning the characteristics of the *PCM*. During the training, the network adjusts to the structure of the *PCM* and forms a topology preserving representation of the manifold. In addition, the learned topology can then be exploited for sensor based path planning. Details of the implemented TRN algorithm are described in Ref. [13].

3.1 Diffusion-Based Path Planning

Based on the topology preserving map of the *PCM*, we generate a path from an initial position to a given target, e.g., to guide an end effector of a robot manipulator in the presence of obstacles within the workspace. For this purpose we 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. The goal is to find a linked chain $i_{0,1,\ldots,n}$ on the graph leading from the current position $i_0 = i_c$ of the end effector to the target position $i_n = i_t$. The complete diffusion algorithm is derived in [12].

After the concentration $f_i(t)$ of the diffusing substance has been calculated for the network topology, a 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 $i_{c,1,2,\ldots,n-1,t}$. The path is short in the sense that it takes a route that maximizes the increase of $f_i(t)$ at each step.

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 [2].

4 Motion Planning for the PUMA

In the following, we will present path planning simulations which were carried out with a general purpose simulator for robot manipulator kinematics and visualization. We also implemented and tested the framework for learning the *PCM* concept on a pneumatic robot arm [12] where accurate positioning presents a challenging problem and can only be achieved by adaptive control methods.

The simulator environment is intended for flexible testing and implementation of neural network control methods, e.g., to test the sensor-based motion planning approach described in this paper. Furthermore, the simulator allows one to define arbitrary camera positions and to visualize the robot motion in 3D. The design of the robot simulator is based on parallel distributed processing and socket-based interprocess communication which allows the different modules of the simulator to run in parallel on a network of workstations. Figure 2 outlines the simulator, together with all its modules and communication paths. The robot controller, in our case a TRN with the submodules for vector quantization, competitive Hebbian learning and diffusion-based path planning, supplies the path plan and the corresponding control signals to execute the plan. The actual robot simulator, on the

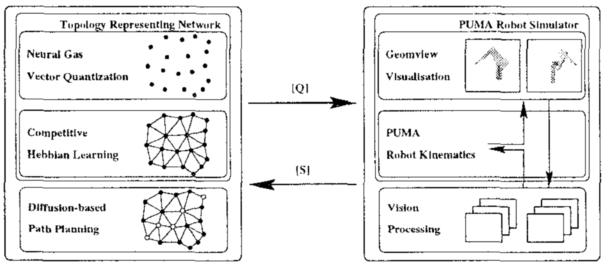

Figure 2: *PUMA robot simulator:* The TRN represents the robot controller and communicates via UNIX sockets with the PUMA kinematics module which handles the data transfer to Geomview, for visualization of the robot arm, and to the vision processing.

right side of Figure 2, includes the kinematic model of the PUMA, the Geomview package as graphical front-end and vision processing subroutines for detecting gripper and obstacles.

The kinematics we employ here is based on the PUMA 562 industrial robotic manipulator, a 6-degree of freedom robot arm with a two-fingered gripper. The forward kinematics for the manipulator is calculated by a PEARL program which also handles the communication to and from the neural network controller as well as the data transfer between Geomview and the vision processing. Visualization of the robot arm is based on Geomview [5], a public domain 3D visualization software[1]. Although it is possible to query Geomview for the position of the robot geometry in 'world coordinates', we chose to use the Geomview camera frames as basis for vision processing. This emulates the situation we encounter for an experimental robotic setup where we would like to avoid any calibration tasks for the kinematic or vision parameters and solely rely on sensor feedback to generate the motion plan. Furthermore, the modular design of the simulator allows the kinematics subsystem to be replaced by a real robot.

4.1 Implementation

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 derived from the Geomview cameras; vision processing resolves the gripper location in each camera frame. Joint position of the manipulator, denoted by $\mathbf{q} = (q_1 \ldots q_n)^T$, is generated by the network during the

[1]Geomview home page:
`http://www.geom.umn.edu/software/geomview/`

learning cycle and concatenated with $\mathbf{s}$ to the input vector $\mathbf{u} = (\mathbf{q}, \mathbf{s})^T$ for the TRN. Following a suitable training period, the topology of the neural network resembles the *PCM*. 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.

To visualize the network dynamics during the training cycle, Figure 3 shows the development of a two-dimensional network. The 4D input vector for the network is generated by concatenating the joint positions q_1 and q_2 with the gripper position $(s_1, s_2)^T$ in one camera frame:

$$\mathbf{u} = (\mathbf{q}, \mathbf{s}) = (q_1, q_2, s_1, s_2)^T \tag{1}$$

At the beginning, all neurons are initialized with random numbers, in this case $\mathbf{w}_i^{in} \in [0, \ldots, 1]$. During the learning process the network is presented with random gripper positions and the vector quantization scheme gradually adapts the input probability distribution. The competitive Hebb rule introduces connections between the units which resemble the topology of the input manifold and, after a suitable training period, the algorithm detects and represents the two-dimensional *PCM* which can then be used to generate a path plan.

Figure 3 shows the network with $n = 150$ neurons projected onto the camera plane in three stages. First, we plot the initial distribution of the weight vectors at $t = 0$ when no connections exist, yet. After $t = 25000$ input vectors, generated by random movement of the manipulator, the network has not yet captured well the input manifold. Several connections are misplaced and the distribution of the weights is not optimal. Finally, after a set of $t = t_{max} = 100000$ training cycles, the TRN covers the complete input space and matches the *PCM* well.

Given this learned representation of the *PCM*, we can use the diffusion algorithm, presented in Section 3.1, to generate a path plan from an initial gripper location to a given target point. Start and desired target location as well as the obstacle are only known in vision space while the path plan is generated by the diffusion algorithm exclusively in *CS*-space to ensure a smooth motion in terms of joint angles. As shown in Figure 3, the robot arm moves from a starting position on the left side of the obstacle to the right side while avoiding the obstacle which can be placed anywhere in the workspace of the robot arm. For static obstacles, we have to run the diffusion process only once and then follow the path. In case of moving obstacles, it is necessary to calculate an updated path at every step.

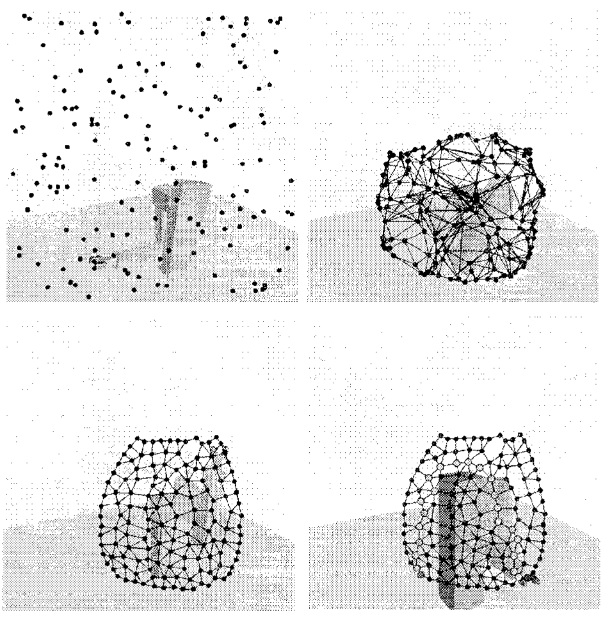
Figure 3: *Development of the topology representing network during the training* projected onto the camera plane. (*top left*) The initial distribution of neurons as initialized at $t = 0$. (*top right*) Intermediate form of the network after $t = 25000$ training moves. (*bottom left*) The final network at $t = t_{max} = 100000$; the latter network resembles well the input manifold. (*bottom right*) Path plan around the obstacle as generated by the diffusion process.

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.

In Figure 4 we plot sample frames from a 3D path, generated by the diffusion process. Here, a network of $N = 700$ neurons has been trained with a data set of 30000 sample moves using three joints of the robot arm and the feedback from two cameras. Therefore, the input vector for the TRN consists of

$$\mathbf{u} = (\mathbf{q}, \mathbf{s}) = (q_1, q_2, q_3, s_1, s_2, s_3, s_4)^T \tag{2}$$

One obstacle is placed in the work space, mapped onto the learned representation of the *PCM* and all connections to neurons which would result in a collision of the gripper with the obstacle are eliminated.

The diffusion algorithm then generates the path and executes it by moving the arm along the planned trajectory. The vision feedback can also be used to correct deviations from the desired path and interpolation techniques can be introduced to further enhance the path resolution [10].

Furthermore, we can impose certain constraints on the *PCM* before generating the path plan with the diffusion algorithm. In Figure 5, we apply a visual constraint by forcing the gripper to be be visible in both camera planes at all times. Therefore, this leads to a different path plan than the trajectory in Figure 4. Other constraints which can be included are kinematic constraints, e.g., limiting the position of one joint to a certain interval.

5 Conclusion

The presented simulations on path control and flexible obstacle avoidance demonstrate the feasibility of our approach for motion planning and illustrate the potential for further robotic applications. Learning the *PCM* with a topology representing neural network introduces a general framework for robot path planning in which sensing, e.g., in the form of camera feedback, is automatically factored into the planning process. The motion plans, thus developed, can exploit properties of the sensed data and can also be linked to appropriately designed vision-based control laws. As our experiments demonstrate, this method can be used to control robotic systems with multiple degrees of freedom in a 3D environment based on sensor data. The proposed 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 of the gripper with detected obstacles.

Acknowledgments

This work was supported by the Roy J. Carver Charitable Trust (M.Z. and K.S.) and by the U.S. Army Research Laboratory under Cooperative Agreement No. DAAL01-96-2-0003 (R.S.). 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. 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.

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

[3] 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.

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

[5] M. Philips. *Geomview: 3D Visualization Software.* 1300 South 2nd St, Suite 500, Minneapolis, MN 55454, U.S.A., 1996. http://www.geom.umn.edu/software/geomview/.

[6] R. Sharma and S. Hutchinson. Motion perceptibility and its application to active vision-based servo control. *IEEE Transactions on Robotics and Automation*, 1997. (to appear).

[7] R. Sharma and H. Sutanto. A framework for robot motion planning with sensor constraints. *IEEE Transactions on Robotics and Automation*, 13(1), 1997.

[8] H. Sutanto and R. Sharma. Global perfomance evaluation of image features for visual servo control. *Journal of Robotic Systems*, 13(4):243–258, April 1996.

[9] R. Y. Tsai. Synopsis of recent progress on camera calibration for 3d machine vision. In *Robotics Review 1.* MIT Press, Cambridge, MA, 1989.

[10] 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.

[11] 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.

[12] M. Zeller, R. Sharma, and K. Schulten. Motion planning of a pneumatic robot using a neural network. *IEEE Control Systems Magazine*, 1997. In press.

[13] M. Zeller, R. Sharma, and K. Schulten. Vision-based robot motion planning using a topology representing neural network. In Jens Kalkkuhl, Ken Hunt, Rafal Zbikowski, and Andrzej Dzielinski, editors, *Applications of Neural Adaptive Control Technology.* World Scientific Publishing, 1997.

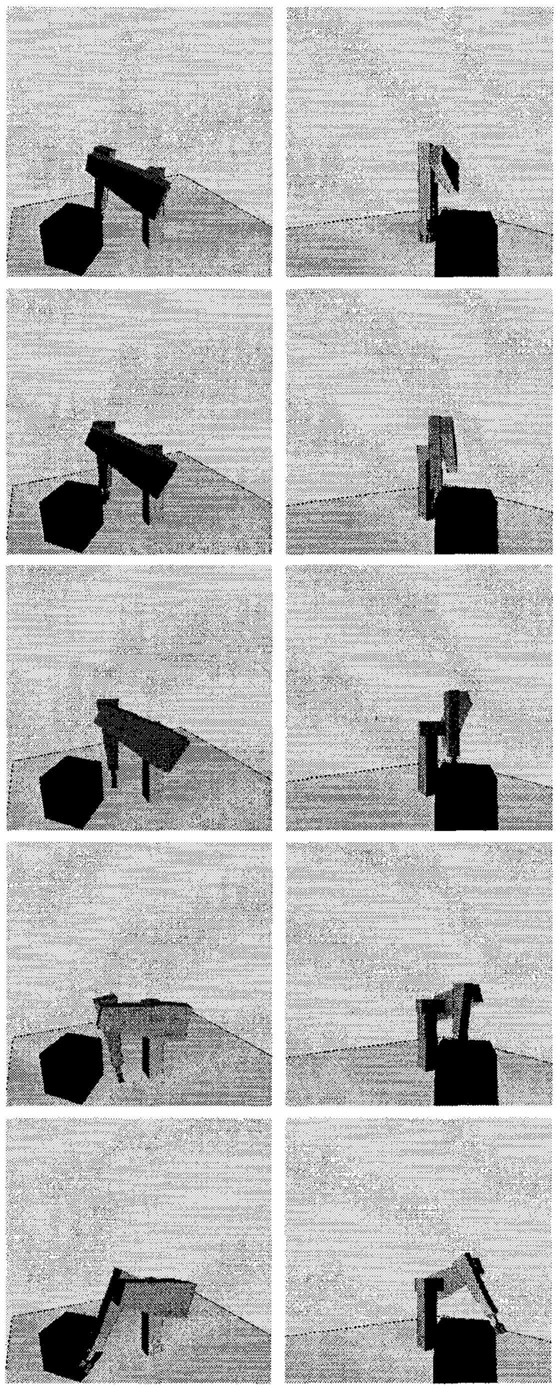

Figure 4: *3D path*, generated by the diffusion process algorithm, as seen by camera 1 (left frames) and camera 2 (right frames). The gripper is sometimes only visible in one camera while being covered by the obstacle in the other camera frame.

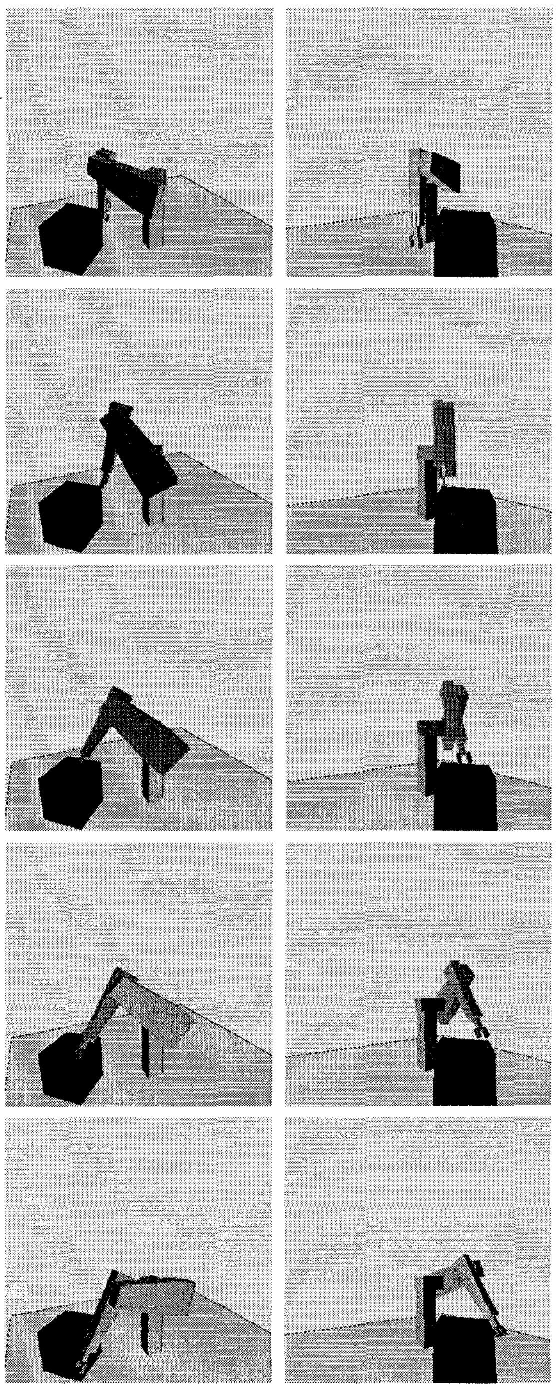

Figure 5: *3D path with visibility constraint:* Camera 1 (right frames) and camera 2 (left frames) for the path. The gripper is visible in both cameras at all times.