Multisensory Visual Servoing by a Neural Network

Guo-Qing Wei and G. Hirzinger

***Abstract*— Conventional computer vision methods for determining a robot's end-effector motion based on sensory data needs sensor calibration (e.g., camera calibration) and sensor-to-hand calibration (e.g., hand–eye calibration). This involves many computations and even some difficulties, especially when different kinds of sensors are involved. In this correspondence, we present a neural network approach to the motion determination problem without any calibration. Two kinds of sensory data, namely, camera images and laser range data, are used as the input to a multilayer feedforward network to associate the direct transformation from the sensory data to the required motions. This provides a practical sensor fusion method. Using a recursive motion strategy and in terms of a network correction, we relax the requirement for the exactness of the learned transformation. Another important feature of our work is that the goal position can be changed without having to do network retraining. Experimental results show the effectiveness of our method.**

***Index Terms*— Cameras, laser range finders, neural networks, visual servoing.**

I. Introduction

When we use robotic external sensors for object manipulation, e.g., grasping, the motion of the robot gripper needs to be determined. Traditional computer vision methods for accomplishing such visual servoing tasks require both the sensors and their mounting parameters to be calibrated in advance [2], [3]. This involves a lot of computations and even some difficulties, especially when different kinds of sensors are used [16], [17]. In the case of multiple sensory visual servoing, an extra difficulty arises in determining the optimal relative weights (importance) of each sensor in the motion estimation procedure (e.g., in the minimization of an objective function [15]). On the contrary, a neural network approach for doing the same job can avoid all such calibrations and provides a convenient sensor fusion framework. The network learns the direct transformation from (multi-) sensory data to the required motions based on a set of examples. The effects of sensor calibration, geometric transformation, and the relative importance of each sensor are all absorbed in a single network.

Kuperstein [7] first proposed to map stereo disparities of stationary cameras directly to the robot joint angles used to reach a single point in the three-dimensional (3-D) space by a nonlinear network. Martinetz *et al*. [9] addressed the same problem by adding a self-organization feature map to achieve a dynamic mapping resolution. Later, Kuperstein [8] considered more degrees of freedom by including orientation information in image features. Miller [10], [11] used an eye-on-hand configuration to track an object on a conveyor. This was done by learning the mapping from the differences between the current and the desired images to the joint angle displacements used to move the end-effector. Since the same relative hand-to-object position in different robot configurations would require different robot joint angle displacements to achieve the desired position, the robot's current joint angles should be involved in the input space, leading to an increase in the dimension of the input space and thus in the computational complexity. Hashimoto *et al.* [4] simplified Miller's method by considering the relative positioning with respect to a *static* object, without having to involve the current joint angle configuration in the input space. The case of static objects, however, is trivial since the desired joint angles are fixed.

A common problem with all previous approaches is that the learned mapping is assumed to be exact in the recall stage, which is in practice either not the case or can only be achieved with increased complexity in learning [4]. Another problem with previous approaches (in the eye-on-hand case) is that the network has to be retrained if one wants to change the desired hand-to-object relative position, because of the change of the mapping.

In this correspondence, we use a multilayer perceptron to learn the direct transformation from multisensory data to the end-effector's Cartesian motion, assuming that the robot kinematics is known *a priori* [14]. The sensors we consider are stereo cameras and laser range finders mounted on a robot hand. We propose a method to relax the requirement for the exactness of the learned mapping by means of network modification and recursive motion. The same network can also be used to adapt to changed goal positions without the need for retraining.

II. The Learning and Control Strategy

A. Learning the Sensor-Motion Transformation

Suppose $S \in R^n$ is the vector of sensor values in the sensor space, where n is its dimension. The sensor value vector S is the input to the network. The general principle to choose the dimension is *uniqueness* and *redundancy*. Here uniqueness means that the sensory data should be able to uniquely determine the end-effector's relative position with respect to the object, while redundancy requires that one use more than the minimum number of sensory data to achieve robustness in the pose determination. In our specific application to be presented in Section III, the sensor value vector contains two image points in each of a stereo pair and four laser ranges. Thus we have the input dimension at $n = 12$ (see Section III for uniqueness analysis). The network output $M \in R^m$ is the Cartesian motion parameters of the end-effector. It is well known that a rigid motion is fully described by a rotation matrix and a translation vector. For the sake of computational savings, it is useful to parameterize the rotation matrix by fewer parameters, e.g., by the representations in [1]. However, we showed [13] that most of those parameterizations are not appropriate to be used as the network output. The reason is that the singularities in the representations break the smoothness requirement for the input-output relationship of the network, which is a precondition for a correct learning and a meaningful generalization [14]. (The smoothness requirement says that smooth changes in the input space should also cause smooth changes in the output space.) We list in Table I the validity of the various parameterizations as the network output. It can be seen from the table that only the roll-pitch-yaw (or the X-Y-Z Euler angles) [1] can be used in a suitable range as the network output. Thus the universal representation of a rotation as the network output is the rotation matrix itself. (But in this case it is difficult to ensure orthonormality of the rotation matrix.) Since the range of motion in our case does not exceed 90° we choose the roll-pitch-yaw as our rotation parameterization for the network output. The output dimension is thus $m = 6$ (three for rotation and three translation).

Manuscript received October 14, 1998. This paper was recommended by Associate Editor M. S. Obaidat.

G.-Q. Wei is with Siemens Corporate Research, Princeton, NJ 08540 USA.

G. Hirzinger is with the Institute of Robotics and System Dynamics, German Aerospace Research Center (DLR), 82234 Oberpfaffenhofen, Germany.

Publisher Item Identifier S 1083-4419(99)02299-2.

TABLE I
THE POSSIBILITY OF VARIOUS PARAMETERIZATIONS AS THE NETWORK OUTPUT

Type of representation	Applicability
Roll-pitch-yaw	$-90° < \alpha, \beta, \gamma < 90°$
$Z-Y-Z$ Euler angles	inappropriate: around the reference position, tiny movements may cause abrupt changes of the α angle
Equivalent angle-axis	inappropriate: around the reference position, tiny movements may cause abrupt changes of the axis of rotation (180°)
Quaternion	inappropriate: the same as the equivalent angle-axis

To obtain the training data, we set the robot end-effector to the desired grasping position, which we call the reference position. Then we make random movement of the robot hand. The amount of motions $\{M_i, i = 1, \cdots, K\}$ with respect to the reference position and the sensory data $\{S_i, i = 1, \cdots, K\}$ after each motion are recorded, where K is the number of motions. The network is then trained by using the backpropagation algorithm [12] with the input–output examples $\{S_i, M_i; i = 1, \cdots, K\}$ as the training data.

B. Dealing with the Inexactness of the Mapping

Theoretically, a feedforward network can approximate any continuous mapping up to any accuracy [6]. But due to the finite number of training samples available and the finite network size used in practice, the actually learned mapping is usually not exact, especially when a large motion range is involved.

Under the assumptions made at the beginning of the last section, there is a one-to-one (bidirectional) correspondence between the sensory pattern and the motion parameters (uniqueness). The transformation from the sensory data to the robot motions is thus monotonic [18]. Therefore, there is only one sensory pattern which induces the zero motion: the sensory pattern S_{ref} at the reference position. If we visualize the mapping in a simplified one-dimensional (1-D) case, there is only one point of intersection of the mapping-curve with the sensor axis S (see Fig. 1). If the learned mapping is accurate enough, the property that there is only one point of intersection with the S-axis in the learned mapping is maintained (see S'_{ref} in Fig. 1). Suppose the exact and the learned mappings are represented by $M = f(S)$, and $M = f'(S)$, respectively. Then, it follows that $f(S_{\text{ref}}) = \mathbf{0}$ and $f'(S'_{\text{ref}}) = \mathbf{0}$. Based on this property, the sensory data S'_{ref} will be called *stable sensory pattern* since it does not trigger any motion; the corresponding robot end-effector position will be called the *stable position.*

Now we analyze the behavior of an *exact* mapping under *recursive motion*. Suppose S_1 is the sensory pattern at an initial gripper position. If the robot moves according to the exact mapping $M_1 = f(S_1)$, the reference position will be reached in one step. With recursive motion (or sensory feedback), we send only a small portion of M_1 to the robot. We denote the small portion by $\Delta M_1 = \rho M_1$, where the constant ρ is used to scale the motion and is made dependent on the magnitude of M_1. After the robot has moved by ΔM_1, we measure the sensory data to obtain, say S_2, at the new position. Then, S_2 is used to recall another small motion ΔM_2. This process repeats until convergence is reached at $M_N = f(S_N) = \mathbf{0}$, where N is the number of motion steps. Since the exact mapping is used here, convergence is guaranteed to be reached at the reference position, that is, $S_N = S_{\text{ref}}$. (Since each step of motion reduces the difference between the current sensory pattern and S_{ref}, the process is convergent.) This procedure can be analyzed by a local linearization method as follows. Since $M = f(S)$ is monotonic, its inverse exists, which will be denoted by $S = g(M)$. Because of the smoothness of

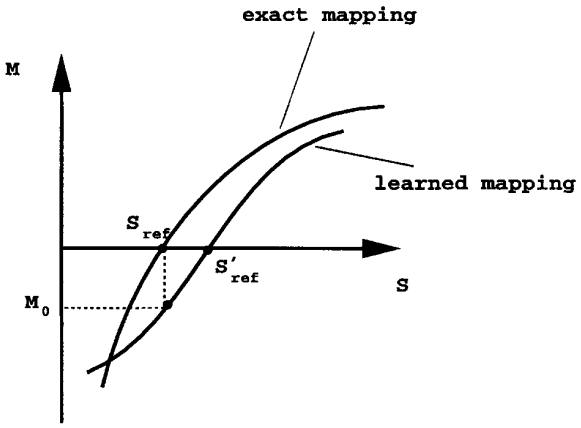


Fig. 1. The exact and the learned mappings.

the mapping $f(\ )$, the inverse function $g\ ()$ is also smooth. That is, a small motion ΔM will cause a small change ΔS for the sensory data. Thus, at the nth step of motion, we have

$$\Delta S_n \approx \frac{\partial g(M_n)}{\partial M_n} \cdot \Delta M_n \tag{1}$$

$$S_{n+1} \approx S_n + \Delta S_n \tag{2}$$

$$\Delta M_{n+1} \approx \rho_n \frac{\partial f(S_n)}{\partial S_n} \cdot \Delta S_n \tag{3}$$

$$M_{n+1} \approx M_n + \Delta M_{n+1} \tag{4}$$

where n indexes the motion step, $n \geq 1; S_n$ is the sensory data at the nth station (i.e., after the $(n-1)$th motion step), M_n is the total amount of movement from the initial position to the nth station; and ρ_n is a scale factor. Therefore, in the case of small motions, the sensor-motion loci are approximately along the mapping curve. It can be seen from the above equations that the dynamic behavior of the recursive motion is completely governed by the differential properties of the mapping, i.e., (1) and (3). Equations (2) and (4) here are only for understanding purposes: The exact value of S_{n+1} should be read from the sensors after the motion ΔM_n. The same applies to M_{n+1}: the total amount of robot motion should be computed from the difference between the actual robot position and the initial position. Based on (1) and (3), if the learned mapping (inexact) is differentially similar to the exact one, the system of the inexact mapping will converge at $M_{N'} = f'(S_{N'}) = \mathbf{0}$, with $S_{N'} = S'_{\text{ref}}$, where N' is the number motion steps. Here the *differential similarity* of the learned mapping to the exact one can be stated more concretely as the sign equality of the derivatives of the learned mapping to those of the exact mapping at every position. (It is the *directions* (signs) of the motions which determine the convergence of the system.) In this way, the requirement for the *value* exactness of the mapping is not necessary. Another important feature of recursive motion is that it ensures a smooth trajectory for an inexact mapping.

C. Network Modification

We have just established the convergence properties of the recursive motion for an inexact mapping. Since the converged position for an inexact mapping is different from the required one (i.e., $S'_{\text{ref}} \neq S_{\text{ref}}$), we have to modify the mapping such that the required reference position be reached. This can be done in two ways. The first is to shift the sensory data in the input space of the network by the amount $S_{\text{ref}} - S'_{\text{ref}}$. The modified mapping reads as $M = f''(S) = f'(S - S_{\text{ref}} + S'_{\text{ref}})$. Since $f''(S_{\text{ref}}) = f'(S'_{\text{ref}}) = \mathbf{0}$, we now have the stable position at $S = S_{\text{ref}}$, i.e., the required reference

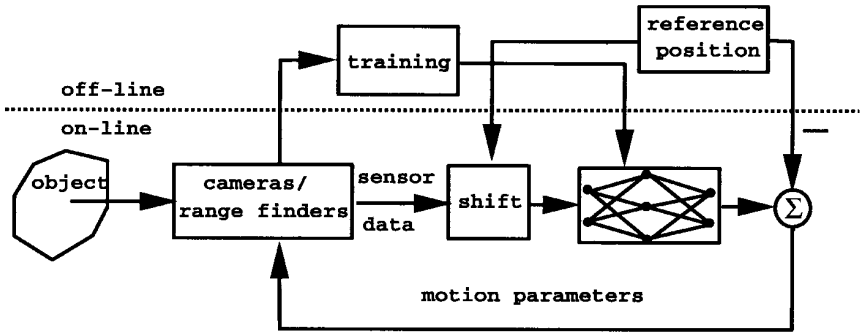


Fig. 2. The system diagram for learning the sensor-motion transformation.

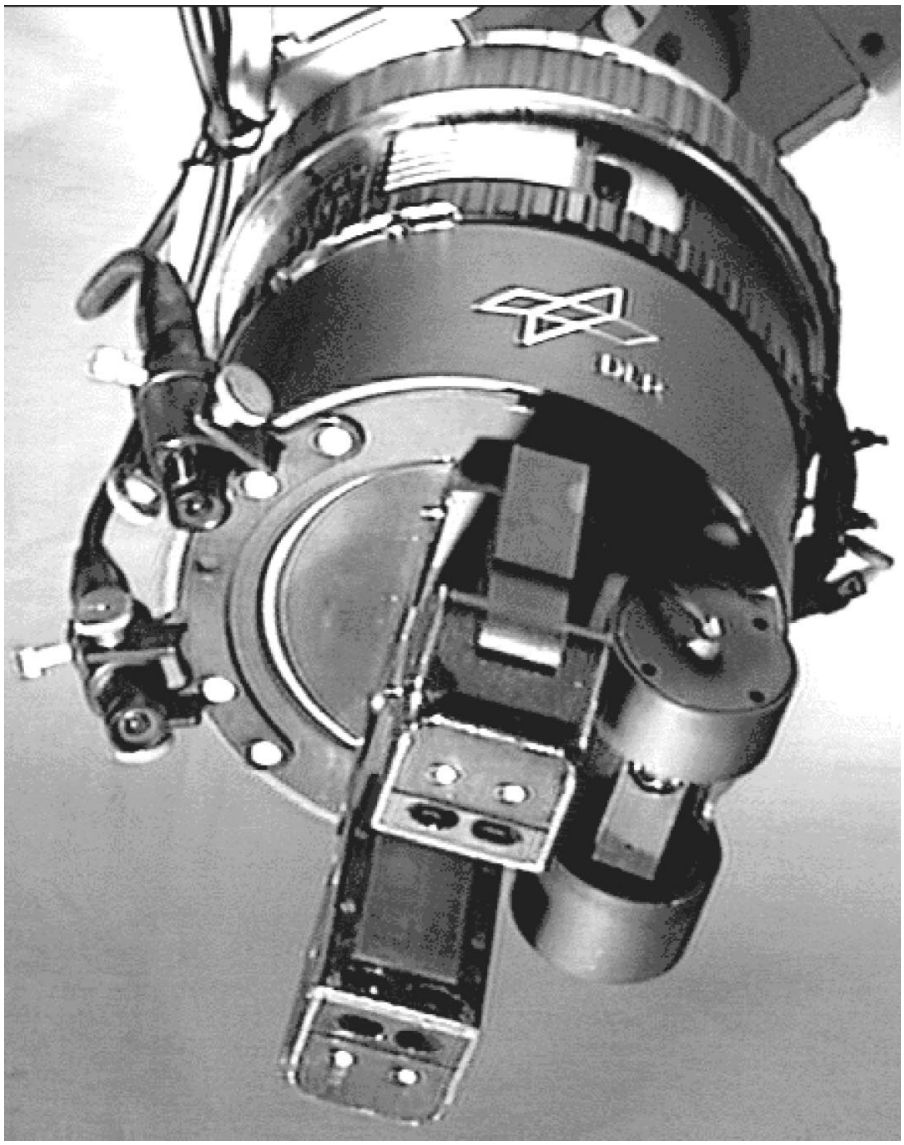


Fig. 3. The multisensory gripper used in the experiment.

position; here, $S'_{\rm ref}$ should be first measured at the converged position by using $M = f''(S)$ in recursive motion. The second method is to shift the network output. We first compute $M_0 = f'(S_{\rm ref})$ using the learned mapping, refer to Fig. 1. Then, the mapping is modified as $M = f''(S) = f'(S) - M_0$, which ensures the stable position to be at $S = S_{\rm ref}$, since $f''(S_{\rm ref}) = f'(S_{\rm ref}) - M_0 = \mathbf{0}$. Since shift in either

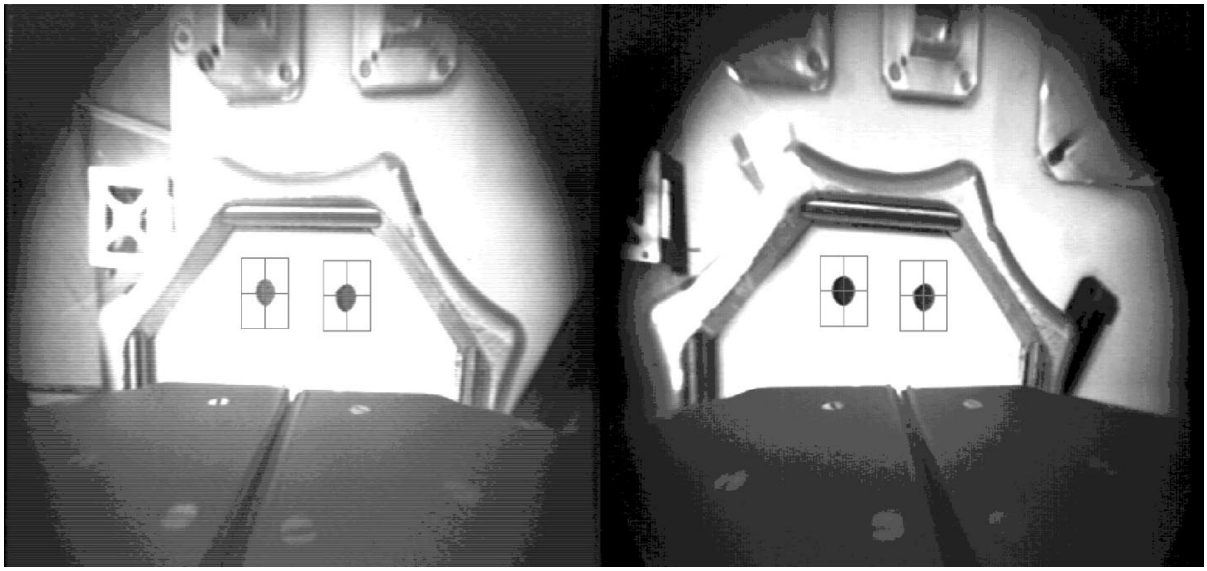


Fig. 4. The stereo image of the object superimposed with the detected centers of gravity and the region of interest in the tracking of the black blobs.

the input or the output space do not change the shape (derivatives) of the mapping, the convergence property of the modified mapping in recursive motion is unaffected.

The above methods can be extended to the case of a changed reference position. Suppose after network training, we want to change the reference position to a new one, which is specified by another sensory data vector S_g measured at that position. We can simply modify the learned mapping according to either of the following formulas:

$$M = f''(S) = f'(S - S_g + S'_{\text{ref}}) \tag{5}$$

and

$$M = f''(S) = f'(S) - f'(S_g). \tag{6}$$

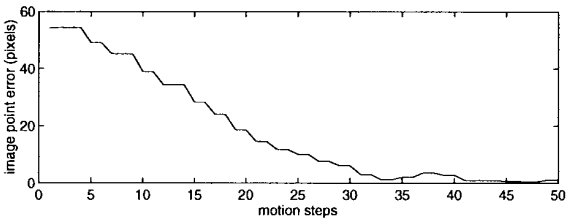
With the modified mapping as either (5) or (6), we have the converged position now at the new reference position $S = S_g$ since $f''(S_g) = \mathbf{0}$. Therefore, we can use the existing network to achieve the new reference position without having to do retraining. Here, it is assumed that the new reference position is within the range where the training data were sampled. Otherwise there is no guarantee that the modified mapping curve has only one point of intersection with the sensor axis in the working range (refer to Fig. 1). Experiments indicate that the use of (5) needs fewer steps of recursive movements than using (6); but to use (5), one has to measure S'_{ref}. Thus we use (6) in our experiment. Fig. 2 shows the complete system-diagram of our method, where the training is done off-line, and the required reference position is used to modify either the network input or the network output.

III. Experiments

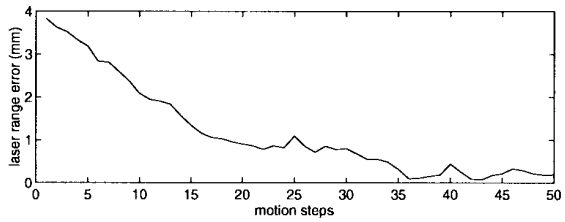
Our method has been tested in a real robot environment. The robot we used is MANUTEC R2. The object we used is called ORU (the Orbital Replaceable Unit), and is chosen from the space robot technology experiment ROTEX in the 1993 German Spacelab-mission D2 [5]. This object is in octagonal shape and is to be grasped in a predefined position by a space robot using telesensor programming. The end-effector is mounted with multiple sensors including two hand cameras and nine range finders (cells) (see Fig. 3). In our experiments, we used 4 of the short range sensors (active range 0–3 cm) mounted on the front side of the gripper and the stereo cameras for visual servoing. When the end-effector is far away from the object, the range finders are blind, and the camera images alone can be used for motion determination. The experiments have been reported in [13]. When the end-effector gets nearer to the object, the object is occluded by the end-effector, so that only two of the corner points on the object can be always seen by the cameras. In this case, the range finders are in work. But neither the camera images nor the range data alone can determine a unique end-effector position, while the use of them both can. The four range finders, hitting on the planar surface of the ORU, determine three degrees of freedom: one positional and two rotational. The camera images determine the other three degrees of freedom by using the two corner points. If we use analytic computer vision methods to solve the same problem, a lot of calibrations have to be involved [15]. The use of the neural network method avoids these computations.

Fig. 4 shows the stereo view of the ORU object in the reference position, where the two black blobs were artificially marked on the object to ease real-time image processing for blob tracking. The center of gravity of the blobs were used as the image feature points. The blob tracking algorithm was realized on a Datacube MaxVideo 200 image processing system, see [15] for details. The detected blobs together with the region of interest for tracking are superimposed on the intensity image in Fig. 4. The absolute accuracy of blob localization by image processing is unfortunately unknown because of the difficult to obtain ground truth. We could only measure the variance of the localization; this is 0.23 pixels. Fortunately, the precision of blob localization is not important in our neural visual servoing method: Any localization error for the blobs can be learned by the network, as long as the error is systematic, e.g., a constant shift. This is not the case for conventional computer vision methods for doing the same job.

A network of size $12 \times 100 \times 6$ was chosen. A set of 250 training samples was randomly generated to cover the work range and used to train the network. Note that due to the relaxation of the exactness of the learned mapping, we do not need to use huge amounts of training data. The conjugate gradient method is used for the training. After 280 cycles of iterations, which took about 5 min on a Indygo2 Silicon Graphics workstation, the training is stopped with a rms error being 4% of the maximum motion component (each



(a)



(b)

Fig. 5. The differences between the current sensor data and the reference sensor data during visual servoing (a) the image data error and (b) the range data error.

TABLE II
THE SENSOR VALUE DIFFERENCES AT THE STARTING POSITION AND THE CONVERGED POSITION

feature index	starting position			converged position		
	ΔX	ΔY	Δd	ΔX	ΔY	Δd
No.1	12.03	23.33	1.75	0.26	0.37	0.125
No.2	13.91	22.30	0.81	0.24	0.27	0.125
No.3	14.22	25.09	1.87	0.74	0.19	0.062
No.4	14.20	22.82	2.72	0.44	0.49	0.041

motion component, i.e., the rotation angles and translation values, is normalized by its own maximum value). The trained network is then used for the visual servoing of the end-effector to any desirable position set within the range of training data without further need for learning. The scaling of motion used in the recursive scheme is done for rotation and translation components separately. For rotation part, we first represent the rotation matrix by the angle-axis form [1]. Then the rotation angle is scaled such that it does not exceed 0.5°. For the translation part, the magnitude of translation is scaled to be within 0.8 mm (if it is larger than this limit). Fig. 5 show the profiles of the differences between the current sensor data, with the gripper started from an arbitrary position, and the reference sensor data during visual servoing for a reference position. (Note that the starting position can be outside the range of the training data.) The sensory data differences at the starting position and the converged one are also listed in Table II, where ΔX and ΔY represent the X and Y image coordinate errors, and Δd the range data error. It can be seen from the figures and from the table that after 50 steps of motion, the average image error and the average range error have been reduced to 0.37 pixels and 0.08 mm, respectively. The result has been much more accurate than that reported in [4], where two networks, one for coarse motion and one for fine motion, were used, and the accuracy in [4] for noise-free (simulated) data was ± 4 pixels.

Concerning the stability of our procedure, occlusions in the sensory data are not allowed. That is, if an image point is out of track, or if a laser range sensor fails, the neural network may misguide the robot. This problem could be solved by training several networks with some sensory data missing (as long as uniqueness is assured), and then switch between the different networks after identifying the type of sensory pattern.

IV. CONCLUSIONS

In this correspondence, we have presented a neural network approach to multisensory visual servoing. A multilayer perceptron network is used to learn the direct mapping from multisensory data to robot motions. We propose to deal with the inexactness of the learned mapping in terms of recursive motion and modification of the network input/output. In this way, an inexact network can be used to achieve the required mapping. This enables computational savings, in the sense that a smaller network can be applied, and fewer numbers of training samples can be used. Another feature of our method is that we do not need to retrain the network if we change the reference position; any position within the training range can be reached by modifying the existing network. Experiments have shown that satisfactory accuracy has been achieved.

REFERENCES

[1] J. J. Craig, *Introduction to Robotics, Mechanics and Control*. Reading, MA: Addison-Wesley, 1986.
[2] B. Espiau, F. Chaumette, and P. Rives, "A new approach to visual servoing in robotics," *IEEE Trans. Robot. Automat.*, vol. 8, no. 3, pp. 129–142, 1989.
[3] J. T. Feddema, C. S. G. Lee, and O. R. Mitchell, "Weighted selection of image features for resolved rate visual feedback control," *IEEE Trans. Robot. Automat.*, vol. 7, no. 1, pp. 31–47, 1991.
[4] H. Hashimoto, T. Kubota, M. Sato, and F. Harashima, "Visual control of robotic manipulator based on neural networks," *IEEE Trans. Ind. Electron.*, vol. 39, no. 6, pp. 490–496, 1992.
[5] G. Hirzinger, B. Brunner, J. Dietrich, and J. Heindl, "Sensor-based space robotics-ROTEX and its telerobotic features," *IEEE Trans. Robot. Automat.*, vol. 9, no. 5, pp. 649–663, 1993.
[6] K. Hornik, "Multipayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, pp. 359–366, 1989.
[7] M. Kuperstein, "Adaptive visual-motor coordination in multijoint robots using parallel architecture," *Proc. IEEE Conf. Robotics Automation*, 1987, pp. 1595–1602.
[8] ——, "Neural model of adaptive hand-eye coordination for single postures," *Science*, vol. 239, pp. 1308–1311, 1988.
[9] T. M. Martinetz, J. J. Ritter, and K. J. Schulten, "Three-dimensional neural net for learning visualmotor coordination of a robot arm," *IEEE Trans. Neural Networks*, vol. 1, no. 1, pp. 131–136, 1990.
[10] W. T. Miller III, "Sensor-based control of robotic manipulators using a general learning algorithm," *IEEE J. Robot. Automat.*, vol. RA-3, no. 2, pp. 157–165, 1987.
[11] ——, "Real-time application of neural networks for sensor-based control of robots with vision," *IEEE Trans. Syst., Man, Cybern.*, vol. 19, no. 4, pp. 825–831, 1989.
[12] *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds. Cambridge, MA: MIT Press, 1986, pp. 318–362.
[13] G. Q. Wei and G. Hirzinger, "Learning motion from images," in *Proc. 11th Int. Conf. Pattern Recognition*, The Hague, The Netherlands, 1992, pp. 189–192.
[14] G. Q. Wei, G. Hirzinger, and B. Bruuner "Sensorimotion coordination and sensor fusion by neural networks," *Proc. IEEE Conf. Neural Networks*, San Francisco, CA., 1993, pp. 150–155.
[15] G. Q. Wei, K. Arbter, and G. Hirzinger, "Active self-calibration of robotic eyes, hand-eye relationships and the application to sensor-based robot positioning," Tech. Rep., German Aerospace Res. Est., Feb. 1995.
[16] ——, "Active self-calibration of robotic eyes, hand-eye relationships with model identification," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 158–166, Feb. 1998.
[17] G.-Q. Wei and G. Hirzinger, "Active self-calibration of hand-mounted laser range finders," *IEEE Trans. Robot. Automat.*, vol. 14, pp. 493–497, 1998.
[18] L. E. Weiss, A. C. Sanderson, and C. P. Neuman, "Dynamic sensor-based control of robot with visual feedback," *IEEE Trans. Robot. Automat.*, vol. RA-3, no. 5, pp. 404–417, June 1987.