A Neural Network Operated Vision-Guided Mobile Robot Arm for Docking and Reaching

Jeremy R. Cooperstock
Department of Computer Science
University of Toronto
Toronto, M5S 1A4 Canada

Evangelos E. Milios
Department of Computer Science
York University
North York, M3J 1P3 Canada

Abstract

A robotic system using simple visual processing and controlled by neural networks is described. The robot performs docking and target reaching without prior geometric calibration of its components. All effects of control signals on the robot are learned by the controller through visual observation during a training period, and refined during actual operation. Minor changes in the system's configuration result in a brief period of degraded performance while the controller adapts to the new mappings.

It is shown that a neural network-based controller can perform rapidly and accurately, taking into account the non-linearities of various mapping functions. Such a controller is easy to train, tolerant of imprecise equipment configurations, and insensitive to camera perturbations following training. This method features real-time adaptivity to changes in mappings, and is simpler than traditional control techniques, which require the solution of the inverse perspective projection and inverse kinematics of the system.

Various operations including *approaching, centering, paralleling, reaching* and *adjusting* are performed by the robot as it navigates towards the target. The robot attempts to grasp targets that are sufficiently close, or approach them while avoiding collisions with obstacles.

1 Introduction

Traditional approaches to robot control using computer vision require that camera images of the scene be converted to real-world co-ordinates, which in turn must be converted to joint angles for the robot arm. Formally, this involves the solution of the inverse perspective projection and inverse kinematics equations of the system. Once this is done, the robot can be controlled very accurately. However, this is generally a very difficult task [11] [5]. If the robot arm or vision system does not behave exactly as described by the perspective or kinematic equations, serious problems may emerge. The alternative is to *learn* the relationships expressed

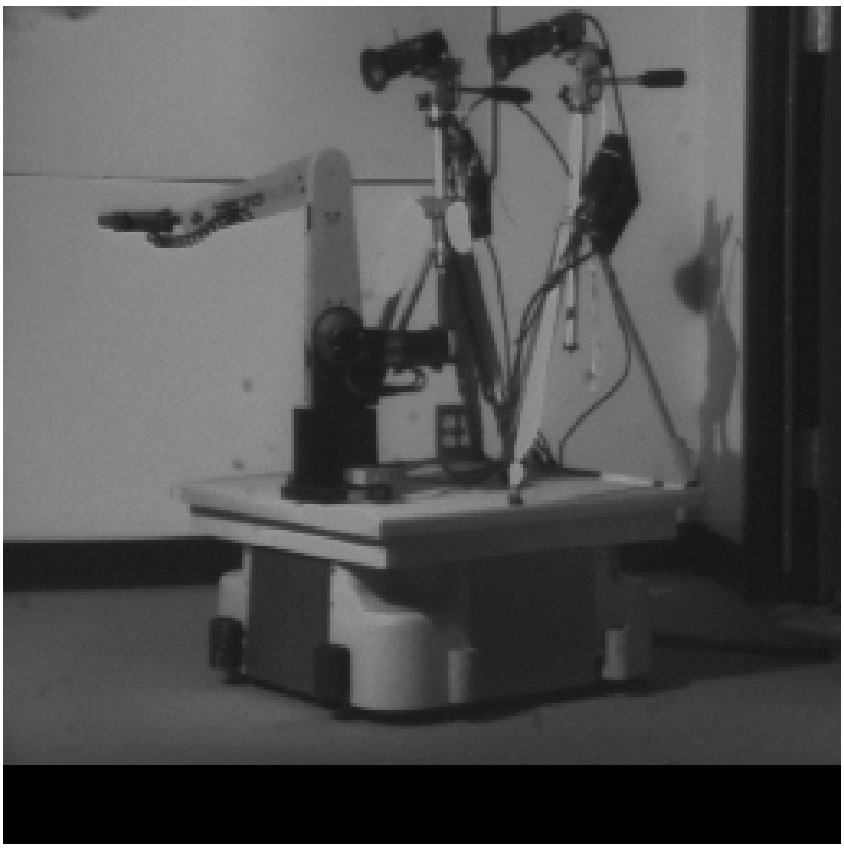

Figure 1: NOVICE consists of a TRC Labmate platform, on top of which are mounted a CRSPlus robot arm, and two Panasonic video cameras. The CRSPlus robot arm which is positioned approximately 400 mm from the camera baseline has segment lengths of 254 mm from shoulder to elbow, 254 mm from elbow to wrist, and 160 mm from wrist to gripper. We use the minimum focal length of 12.5 mm for both cameras, which provides a field of view of approximately 30^0 horizontally.

in these equations by observing the interactions of the robot with the environment.

We propose a relatively simple robot controller, which learns by observing its performance to perform docking and target reaching using no sensing besides low-level vision. Our robot is named NOVICE, for **N**eurally **O**perated and **VI**sually **C**ontrolled **E**quipment (see Figure 1). Motor commands are issued by a controller, made up of a set of independent neural networks using video image co-ordinates as input. As opposed to a conventional model which calculates the inverse perspective projection of the target image and then applies the necessary inverse kinematics, our controller adjusts the direction and position of the platform and the posture of the arm without explicitly computing these transformations. The tasks that NOVICE performs are to dock at a table with a target object on it, and reach for that object. NOVICE may have to navigate around the table to get to a position from which the object is reachable.

The remainder of this paper is organized as follows: In section 2, we review previous work in the area of

sensor-guided robot control. In section 3, we present our design methodology for the neural controller. Section 4 formally describes the tasks that NOVICE performs, providing a context for the discussion of controller architectures in section 5. There, the neural network and geometric methods of control are compared. Section 6 deals with the operation, and section 7 with the adaptability of the entire system. Finally, in section 8, we draw several conclusions from this research.

2 Previous Work

Puget and Skordas [13] propose a "state of the art" technique for traditional robot control involving camera calibration through the explicit computation of a transformation matrix and minimization of a mean square error. To extend this technique to docking and reaching, the camera parameters must be found and the inverse perspective projection problem solved to provide the position of the object in three-dimensional space. Clearly, this is not an easy task. However, neural network methods provide a direct mapping from images to motor signals, thereby avoiding the inherent problems of calibration and the associated numerical sensitivity.

Kuperstein and Rubinstein [9] describe a neural controller called INFANT which uses stereo camera views of a target to control four non-redundant joint angles of a robot arm. Another neural control method, used by van der Smagt and Kröse [14], consists of a monocular system which learns and uses inverse differential kinematics exclusively. While this proved effective for control of one joint in a two-dimensional workspace, we found that it does not scale well to multiple joints in three-dimensional space. Instead, we propose that this technique be used for making small adjustments only after the end effector of the robot arm has been positioned near the target.

Martinetz et al. [10] and subsequent researchers [1] have developed topologically ordered neural maps based on Kohonen's work [8] to learn the relationship between positions in image co-ordinates and joint angles of a robot arm. Impressive accuracy is obtained in simulation results but these require large training sets of thousands of samples which may be costly to generate for actual robots. In contrast, our emphasis is on simple networks which can perform reasonably accurately in the real world using very small training sets.

3 Design Methodology

Several design criteria are followed throughout the implementation. NOVICE should be easily trainable after a reconfiguration of the camera positions, a change in the focal characteristics of the lenses, or even a change of the arm or platform. No knowledge of the camera characteristics nor of the arm geometry is known to the controller before training begins. All such information is learned by the controller through observing the effects of a set of control signals to the robot.

We wish to avoid the problems of classical calibration, including the need for explicit geometry measurements, the inaccuracies resulting from non-linearities in the equipment, and the inability to respond rapidly to alterations in the camera configuration [13]. To further this goal, NOVICE does not explicitly compute any distance measurements. Instead, all computation is performed implicitly by a set of neural networks which make up the controller.

The neural network approach provides a uniform treatment of non-linear mappings [7] between input sensory data and output control signals and can adapt to changes in the mapping through time. If a constant, linear relationship exists between input and output, a simple geometrical scheme could be used instead. However, in the real world, the mappings are generally time variant and non-linear as a result of distortion and parameter drift [2]. In order to capture such mappings, an adaptive, non-linear scheme is required. Numerical methods such as those of [4] and [13] could be used instead, but such techniques are in general numerically unstable and require intensive computation, completely at odds with our requirements for a simple yet robust system.

3.1 Terminology

An example sketch of the camera images is provided in Figure 2. The symbols in these images, used throughout this paper are defined as follows:

G	gripper position
T	target position
E	table edge position
S	table edge slope
M	middle (center) of image

These symbols may take the subscripts L and R to denote left and right camera images, and X and Y to denote the x and y rectangular co-ordinates, respectively.

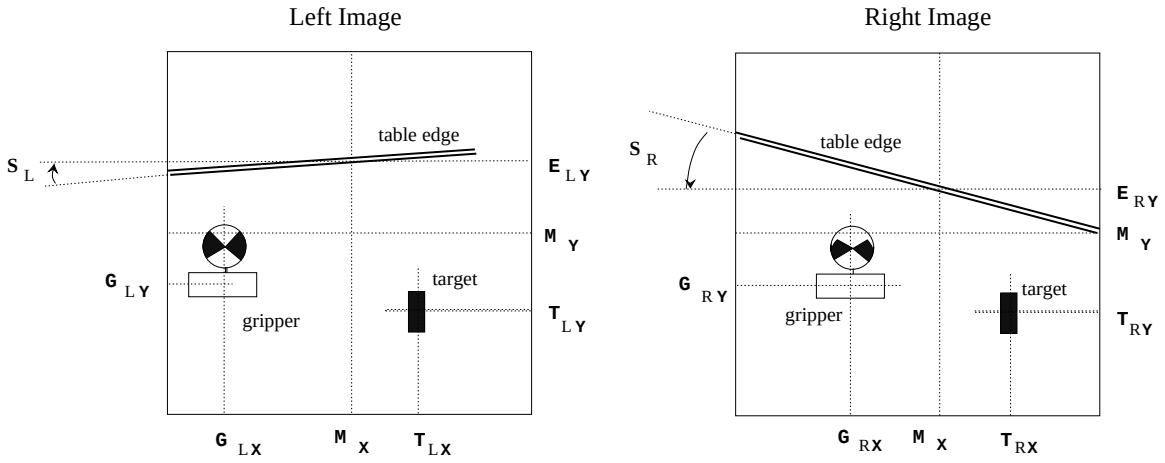

Figure 2: Symbols used throughout this paper to refer to various image objects.

We define two functions, $\mathcal{C}$ and $\mathcal{D}$ to be the *current* and *desired* values of any symbol described above. If no function on a symbol is specified, the current value is assumed.

Finally, we define two terms, *image space* and *joint space* as follows:

Definition 3.1 *Image space is the set of all possible x and y co-ordinates of an object in the left and right camera images, measured in pixels. Joint space is the set of all possible combinations of rotor, shoulder, and elbow joint angles of the robot arm, measured in either degrees or radians.*

4 System Tasks

NOVICE performs one translation operation, approaching; two turning operations, centering and paralleling; and two arm operations, reaching and adjusting. High-level task co-ordination is currently hard-wired in NOVICE. A conventional algorithm, described later, determines the applicability of each operation, either unilaterally, or by consulting the individual neural networks. When an operation is necessary, NOVICE passes control to the appropriate neural network. The remainder of this section describes the various tasks in more detail.

4.1 Approaching

If NOVICE is not sufficiently close to the target to permit reaching, approaching is necessary. Approaching consists of moving forward a fixed amount, so long as this does not result in a collision with the table. Collision avoidance must be handled by visually obtained information, in this case, the height of the table edge, E_Y,

in the image planes.[1] From this value, the controller decides whether a further approach is possible without collision.

4.2 Centering

Centering involves rotating the platform until the target can be reached by moving the robot forward. This is necessary to keep the target visible during subsequent operations as well as to minimize the path length that NOVICE must travel during an approach.

Definition 4.1 *Center error is the signed difference of the horizontal target position, T_X, and the horizontal center of the image, M_X, measured in pixels.*

The inputs to the controller are the center errors from the two images and the output is an angle of rotation for the mobile platform. Centering minimizes the sum of the center errors from the left and right images.

4.3 Paralleling

The paralleling operation is required to bring the platform parallel to a horizontal straight edge, so that NOVICE can continue its navigation along that edge. This enables NOVICE to follow the perimeter of the table to arrive at a position from which the target is reachable. To accomplish this, the platform is positioned so that the direction of forward motion is perpendicular to the horizontal edge and then rotated 90^o.

Definition 4.2 *Slope error is the signed difference between the current slope, $\mathcal{C}(S)$, and desired slope, $\mathcal{D}(S)$, of the image of the horizontal edge of an obstacle.*

To perform this operation, the controller takes the slope error, $\mathcal{C}(S) - \mathcal{D}(S)$, from the two images and provides an angle of rotation to the platform. Paralleling the platform involves minimizing the sum of the slope errors of the obstacle edge. Once the vector of platform motion is perpendicular to the obstacle, NOVICE determines which heading will minimize the distance to a corner and then turns 90^o in that direction.

[1] In our current configuration, one edge is highlighted. While this is a minor modification of the environment, it greatly simplifies the task from a vision perspective.

4.4 Reaching

Reaching consists of extending the arm so that the gripper can be closed around the target. Visual feedback is used to improve the position of the arm when required. The target co-ordinates in the left camera image, T_{LX} and T_{LY}, and the right image, T_{RX} and T_{RY}, are used as input to the controller, which then provides joint angles for the robot arm as output. We consider a target to be *reachable* if the controller's prediction of the required arm joint angles corresponds to a valid posture, that is, if all of the arm signals are within their bounds.

Definition 4.3 *Reaching error, measured in pixels, is defined by the Euclidean distance*

$$\sqrt{(\mathcal{C}(G_X) - \mathcal{D}(G_X))^2 + (\mathcal{C}(G_Y) - \mathcal{D}(G_Y))^2} \tag{1}$$

where $\mathcal{C}(G_X)$ and $\mathcal{C}(G_Y)$ are the current image co-ordinates and $\mathcal{D}(G_X)$ and $\mathcal{D}(G_Y)$ are the desired image co-ordinates of the gripper.

Since the accuracy of the system is dependent upon the focal length of the camera lenses, using pixels rather than millimeters provides a more consistent measure of reaching error. Although NOVICE measures errors internally using pixel values, we will sometimes display our results in millimeters. Note that occasionally, a decreased error in image space corresponds to an increased error in joint space. This is discussed in greater detail in [3].

By training the reaching controller near the bounds of the arm's joint angles while the gripper is still visible, NOVICE can later predict whether or not a target is reachable by running the system forward on the target and testing that none of the resultant output signals exceeds its legal range. If the target is reachable, the controller directs the arm to assume the appropriate posture. Otherwise, the controller decides whether to approach further, or, if NOVICE is in danger of crashing into the table, to navigate around its perimeter.

4.5 Adjusting

Adjusting involves making small changes to the current joint angles of the arm so as to bring the current gripper position closer to its desired position. It is desirable that this process converge rapidly, and that the gripper not oscillate near the desired position.

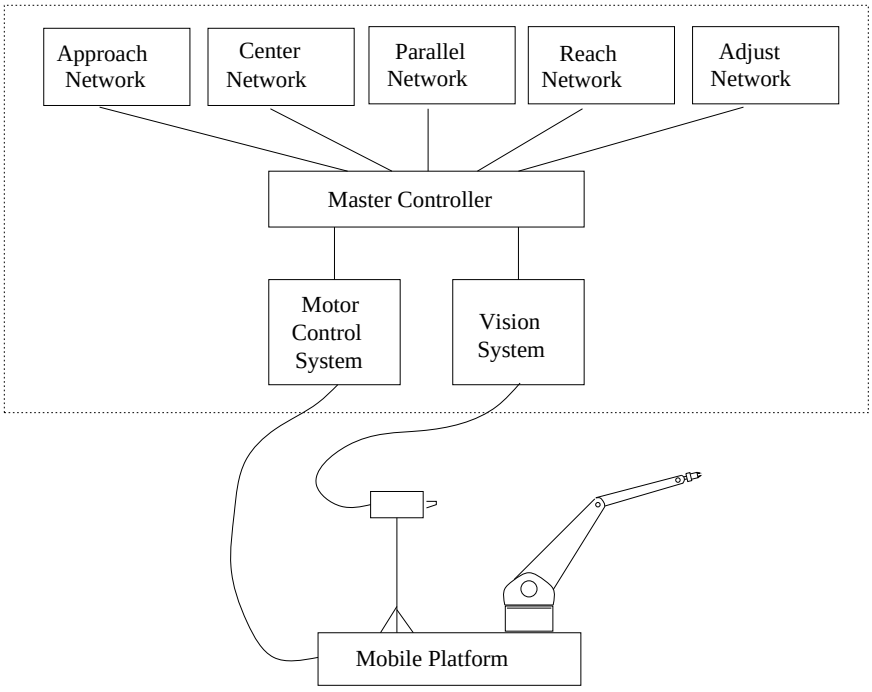

Figure 3: The NOVICE controller consists of five feed-forward neural networks used for the operations of *approaching, centering, paralleling, reaching*, and *adjusting*. The master controller relays input signals from the vision system to the various networks and output signals from the networks to the motor control system.

5 Controller Architecture

The NOVICE controller, shown in Figure 3, consists of five feed-forward neural networks, a vision system, and a motor control system, which are all arbitrated by a master controller. Each network maintains a set of the previous examples seen from which randomly selected tokens are continually used for training.

All weights are initialized to random values between 0 and 0.2 and each neuron's output is determined by the sigmoidal activation function:

$$\frac{1}{1 + e^{-\sum_{i=1}^{n} input_i + bias}} \tag{2}$$

where $input_i$ represents the ith input potential and $bias$ is the unit's bias level. To produce a value of 0 or 1, the sigmoidal function requires that the sum of the inputs be $-\infty$ or ∞ respectively. Thus, as the desired output value of a unit approaches 0 or 1, the link weights into that unit must grow to very large magnitudes. To keep the weights within reasonable bounds, we do not use the full range (0 to 1) of the activation function. Instead, the minimum and maximum output values allowed are 0.25 and 0.75 respectively. With reasonably small weights, the network maintains a greater capacity to adapt its performance to new data. Generally, this

results in significant performance improvements [6].

Minimization of network error is achieved by adjusting the weights with the standard back-propagation algorithm using the generalized delta rule [15]: For any output unit, j, the error signal can be calculated as

$$\delta_j = (t_j - o_j)\ o_j\ (1 - o_j) \tag{3}$$

and for any hidden unit, j,

$$\delta_j = o_j\ (1 - o_j)\ \sum_k \delta_k\ w_{kj} \tag{4}$$

where t_j is the desired output, o_j is the actual output, and w_{kj} is the weight of the kth output link of unit j.

After the nth presentation of input data, the weights of the links between any two units j and i are adjusted by

$$\Delta w_{ji}(n+1) = \eta(\delta_j\ o_i) + \alpha \Delta w_{ji}(n) \tag{5}$$

where η is the learning rate and α is the momentum constant. The values used here are $\eta = 0.3$ and $\alpha = 0.9$.

A sequence of platform motions and arm manipulations is required to train the controller. Presently, this process takes approximately twenty minutes, due largely to the bottleneck in image analysis. If the configuration has not been modified since the last run, previous training data can be loaded instead. As each data sample is obtained, the master controller sends the information to the appropriate neural network, which stores it in a set of training data. NOVICE continuously improves the mappings of each neural network by randomly selecting items from the training sets, running the networks forward, and then back-propagating the errors.

The remainder of this section describes the construction and training of the individual networks in more detail as well as some alternative methods.

5.1 Approaching

The architecture for this network is very simple, consisting of two input units, three hidden layer units, and one output unit. The input units record the table edge height, E_{LY} and E_{RY} at at the horizontal center, M_X, of each image and the output unit provides a binary value indicating whether or not NOVICE can make a further approach without colliding with the table.

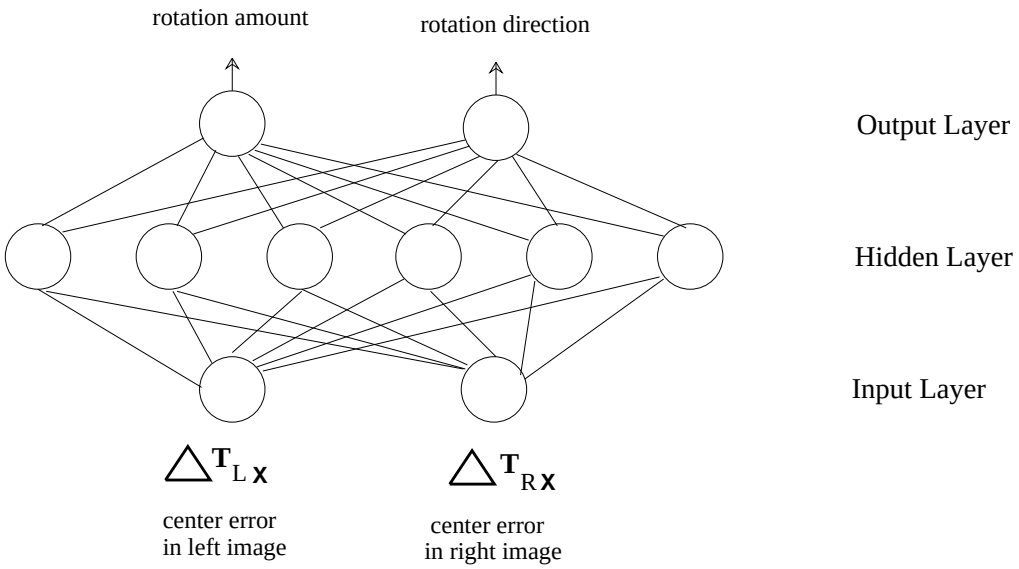

Figure 4: Centering network using pre-processed input.

Prior to training, NOVICE is positioned as close to the table as possible and the camera views of the table edge are assigned a binary label of *legal*. NOVICE then drives towards and away from the table at various angles, assigning labels of either *legal* or *illegal* to the camera views, based on odometry measurements. The approaching network is then trained to correlate the table edge height in the camera images with these corresponding labels. This method allows NOVICE to learn to make the correct decision for approaching regardless of its orientation with respect to the table, provided it is trained with appropriate data. A traditional method relying on simple thresholds of the table edge height would not provide this flexibility.

5.2 Centering and Paralleling

5.2.1 Pre-processed network

The architecture for the turning operations uses two inputs which are given pre-processed data: the difference between the current and desired values of either center or slope error. The network is trained to correlate these values with rotation angles of the platform. Figure 4 provides an example of this architecture for the centering network. Rather than having the network learn the relationship of current and desired center or slope errors to rotation angles, we first compute the difference between these errors and use this value as the input. For example, the inputs to the centering network are $\Delta T_{LX} = \mathcal{C}(T_{LX}) - \mathcal{D}(T_{LX})$ and $\Delta T_{RX} = \mathcal{C}(T_{RX}) - \mathcal{D}(T_{RX})$.

During operation, the desired target position, $\mathcal{D}(T_X)$, is always considered to be the center of the image,

M_X, and the desired edge slope, $\mathcal{D}(S)$, is always zero. However, during training, the actual values $\mathcal{C}(T_X)$ and $\mathcal{C}(S)$, observed in the images are used.

5.2.2 Alternative methods

An alternative to neural network control is to simply calculate the ratio of the change in center or slope error to rotation of the platform, assuming linearity and stability of this value. Linear interpolation could then provide reasonably accurate values of rotation angle for future turning operations. We therefore refer to this method as linear interpolation of geometry measurements. Although this method is trivial to implement, it's performance was seen to be more sensitive to target or table edge proximity than the neural network method.

Another possible architecture for the turning operations has an array of 15 units in the output layer, each unit representing a different amount of rotation [12]. The desired activation for each output unit x_i is determined by the following Gaussian equation:

$$x_i = e^{-\frac{d_i^2}{10}} \tag{6}$$

where d_i is the distance of the ith unit from the output unit indicating the correct rotation amount. The motivation for expanding the output layer this way is that in theory, with the increased number of adjustable weights, the network can fine-tune its output for improved accuracy. However, in practice, the Gaussian network did not perform noticeably better than the pre-processed network.

5.3 Reaching

5.3.1 Traditional Inverse Kinematics Method

The traditional approach to robot arm control consists of solving the inverse kinematic equations for a particular system at a given point in space. In order to control such a system with visual sensing, the inverse perspective projection problem must first be solved in order to map camera image co-ordinates to three-dimensional space co-ordinates. Ignoring the issues of numerical sensitivity, the difficulties in obtaining accurate measurements, and the computing power required to calculate the necessary joint angles, this method is impractical for many systems because of its non-adaptivity to changing configurations. If a camera is perturbed or the robot arm replaced, the kinematic or perspective projection parameters may have to be recalculated off line,

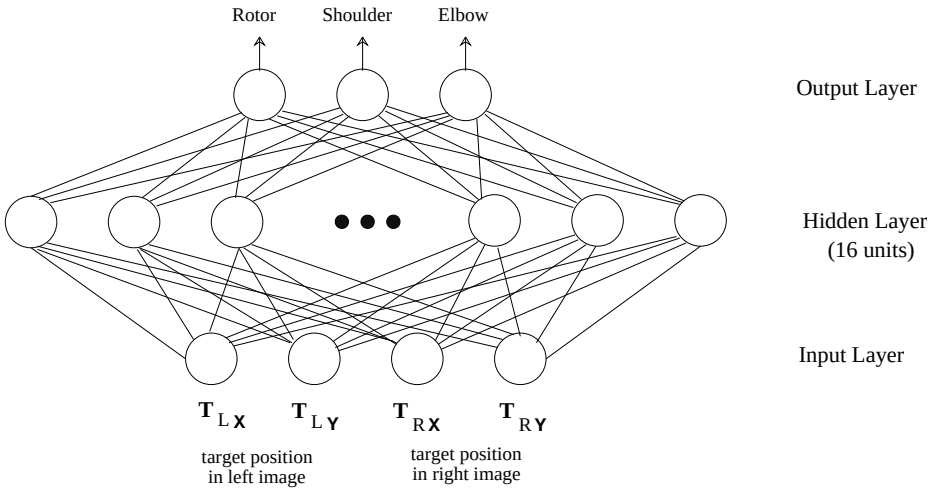

Figure 5: Reaching network.

usually an expensive task.

5.3.2 Neural Network Method

Our network, shown in Figure 5, uses four input units (horizontal and vertical position of the target for each of the two images), 16 hidden layer units, and three output units (base rotor, shoulder, and elbow angles). Unlike [9], NOVICE is not given the disparity value between the two image views of the target. Instead, disparity is computed implicitly by the network. Improvements in performance are possible by increasing the number of hidden layer units to a certain point at the expense of increased training time. It is also possible to construct the output layer with Gaussian response units, simply by replacing each output unit in the original network with an array of Gaussian units. However, the minor improvement in reaching accuracy does not seem to justify the additional training time required. We settled on this particular architecture as a compromise between accurate performance and speed of adaptation to new data.

Many of the possible arm configurations place the gripper outside the field of view of the cameras. Thus, it would be inefficient to train the network by observing the results of a series of randomly chosen postures. Also, pre-computing the range of joint angles which correspond to visible gripper positions would entail calibration and knowledge of arm geometry, both in violation of our design methodology. Instead, the reaching network is trained by moving the arm through the entire range of shoulder and elbow angles. At each position, the rotor angle is randomly selected between a left and right extreme[2] so that training samples are well distributed

[2]With the configuration we are using, the left and right extremes of the rotor angle are -25° and 25°, respectively.

training examples	center	left	right	top	bot	top left	bot right	mean
8	16	11	30	29	12	35	19	21.7
22	11	27	18	8	11	22	17	16.3
63	5	16	17	16	9	10	15	12.6
143	20	3	8	17	8	10	16	11.7
mean	13	14	18	18	10	19	17	

Table 1: Reaching error (in pixels) for different target locations appearing in the left and right camera images. Note that for our configuration, an error of 15 pixels corresponds to a physical error of approximately 10 mm on average.

across the image planes without exceeding the left or right edge of the images. The reaching network is trained to correlate the camera views of the gripper with the corresponding set of joint angles. The training method, described in more detail in [3], obviates the need for arm calibration and reduces the amount of time in which the gripper is not visible.

5.3.3 Performance of reaching network

The accuracy of the reaching operation for various training set sizes and target locations in image space is compared in Table 1 and Figure 6. Note that reaching errors less than 10 pixels were difficult to measure accurately in three-dimensional co-ordinates as the gripper is essentially contacting the target. Reaching accuracy, on average, tends to be best near the center position of image space, (M_X, M_Y), and worst towards the corners. Intuitively, this is sensible, as the center of image space is also the approximate center of the training data distribution.

5.3.4 Error sources

Potential sources of error include neural network generalization inaccuracies, as well as quantity, reliability, and density distribution of training data. However, it is important to determine the effect of uncertainty in the visual estimation of the target position on the resulting gripper position. As shown in the appendix, a single pixel error along the horizontal axis of one camera image can result in an error in target position estimation on the same order of magnitude as the reaching errors exhibited by NOVICE.

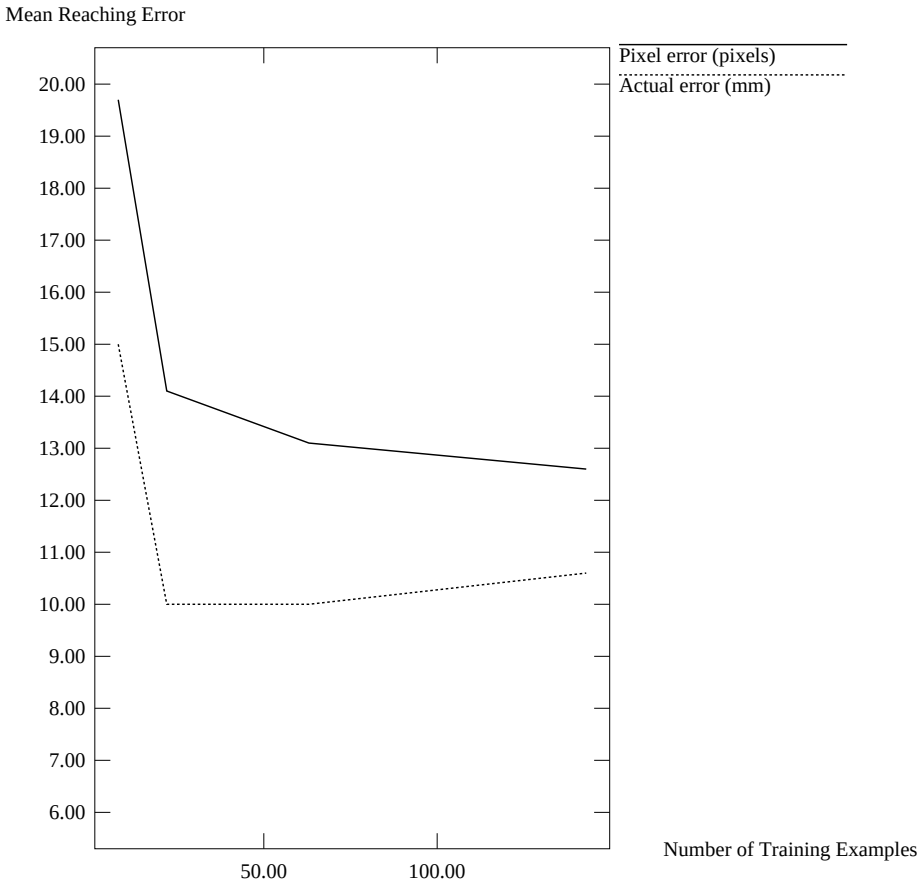

Figure 6: Reaching error for different training set sizes of the reaching network.

5.4 Adjusting

Initially, NOVICE exhibited sufficiently accurate reaching so that gripper position adjustment was unnecessary. However, we wanted a robust system which could deal with inaccuracies of the mapping, possibly as a result of noise, inadequate training, or camera perturbations.

5.4.1 Heuristic Method

NOVICE obtains the additional ability to adjust the gripper position as a consequence of learning the reaching operation. Thus, we can simply make use of the mapping already learnt by the reaching network to make the necessary positional corrections as an alternative to constructing and training an additional network for this task. Kuperstein [9] realized that the accuracy of the reaching network could be improved by training it with new data samples obtained during successive reaching attempts. This technique can be taken one step further by actually *correcting* the gripper position with a heuristic method.

Referring to Figure 7, $\mathcal{C}(G)$ is the current gripper position and $\mathcal{D}(G)$ is the desired position in image space. The joint angles associated with these two positions are $\overrightarrow{\Theta_C}$ and $\overrightarrow{\Theta_D}$, respectively.

After observing the gripper position $\mathcal{C}(G)$ to be in error from $\mathcal{D}(G)$, the reaching network is asked to provide a "guess", $\overrightarrow{\Theta_G}$ for the joint angle measures that correspond to $\mathcal{C}(G)$. We denote the image position of the gripper corresponding to these estimated angles as $\mathcal{G}(G)$ (for "guess" of gripper position). If the reaching errors are small, then so are the required joint angle changes. Furthermore, with a relatively well-distributed training set, the reaching network develops a smooth mapping function with reasonably constant error vectors in a given neighborhood of image space. Under these conditions, we may assume approximate linearity in the relationship between changes in joint angles and image co-ordinates of the gripper, even though this relationship is actually non-linear[3] Thus, the difference between the joint angles $\overrightarrow{\Theta_G}$ and $\overrightarrow{\Theta_C}$ should be approximately equal to the difference between $\overrightarrow{\Theta_C}$ and $\overrightarrow{\Theta_D}$. Using this result, we obtain the following equations for the joint angle vectors:

$$\overrightarrow{\Theta_G} - \overrightarrow{\Theta_C} = \overrightarrow{\Theta_C} - \overrightarrow{\Theta_D} \tag{7}$$

$$\overrightarrow{\Theta_D} = \overrightarrow{\Theta_C} + (\overrightarrow{\Theta_C} - \overrightarrow{\Theta_G}) \tag{8}$$

[3]Note that this assumption relates entirely to properties of the neural network mapping and not to the inverse kinematics of the robot arm.

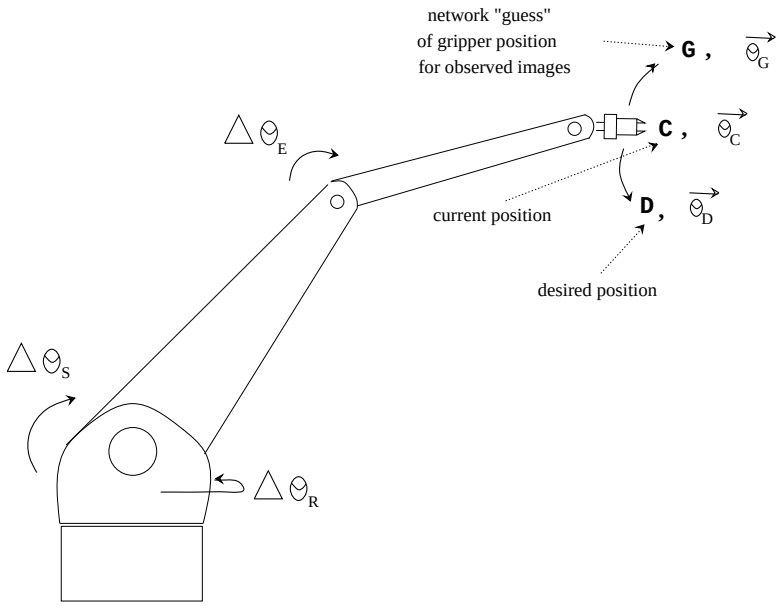

Figure 7: The heuristic method involves using a network "guess" of the current gripper position to make adjustments to the joint angles.

If this first attempt at correcting the gripper position does not succeed with sufficient accuracy, NOVICE can repeat these calculations to obtain convergence to the desired position, $\mathcal{D}(G)$. The convergence rate can be increased by retraining the reaching network with data obtained during our attempts at correcting the gripper position.

5.4.2 Neural Network Method

Although a network which maps current and desired state to a set of joint angle increments [14] is not suitable for initial positioning of the gripper, it is a valid method for making positional corrections where such corrections are small. In this case, the mapping is essentially the inverse differential kinematics of the robot arm. To implement the adjusting network for NOVICE, we require a total of eight input units, four to encode the current gripper position in each image[4] and four to encode the desired change to the gripper position (see Figure 8). The output layer is identical to that of the previously described reaching network, except that each unit describes an increment to a joint angle rather than the joint angle itself.

To train the adjusting network, the three joint angles set during training of the reaching network are

[4]It could be argued that only three inputs are required to encode gripper position, since a vector of joint angles can be substituted for the image position values. However, the rest of our design involves learning a mapping directly from image coordinates to joint angles. Therefore, such an encoding would be inconsistent with our methodology, and only result in minor savings at best.

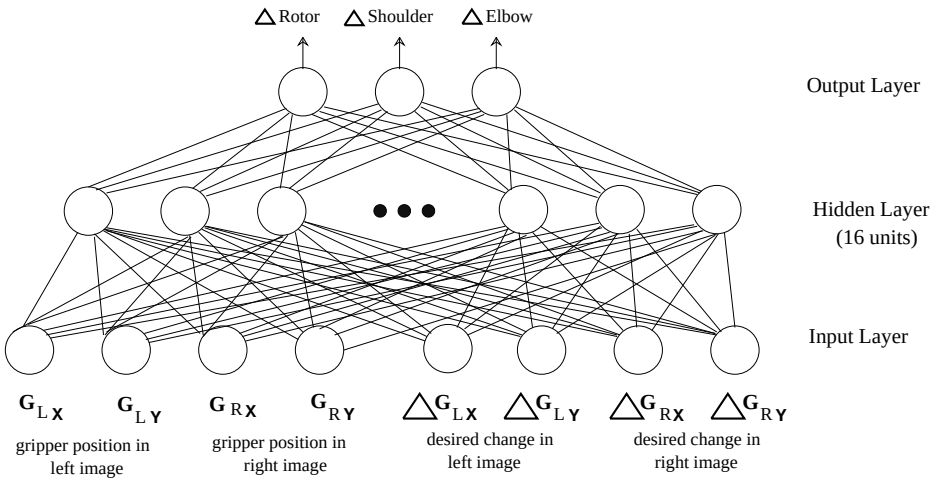

Figure 8: Adjusting network.

randomly perturbed, and the resulting change in gripper position observed. This process can be repeated several times if more data is required. We note that the adjusting mapping is reversible. If a perturbation of $\overrightarrow{d\Theta}$ to the joint angles results in a movement from position P_1 to P_2, then a perturbation of $-\overrightarrow{d\Theta}$ to the joint angles results in an opposite movement from P_2 to P_1. This realization allows us to obtain two samples of training data for every perturbation of the joint angles.

5.4.3 Comparison of adjusting methods

To compare the two adjusting methods, the reaching network training data was corrupted by randomly perturbing the joint angles corresponding to each training sample by $\pm 5\%$. This ensures that the reaching network will no longer provide sufficient accuracy to achieve grasping on the first attempt. The results are shown in Figure 9. Although the adjusting network outperforms the heuristic method for sufficiently large training sets, it is important to note that the heuristic requires no additional training beyond the reaching network. Furthermore, the heuristic offers the advantage of corrective movements that are proportional in magnitude to the reaching error. In contrast, the adjusting network learns to make corrections no greater than the largest magnitude used during training. Thus, its performance is only guaranteed for initially small errors.

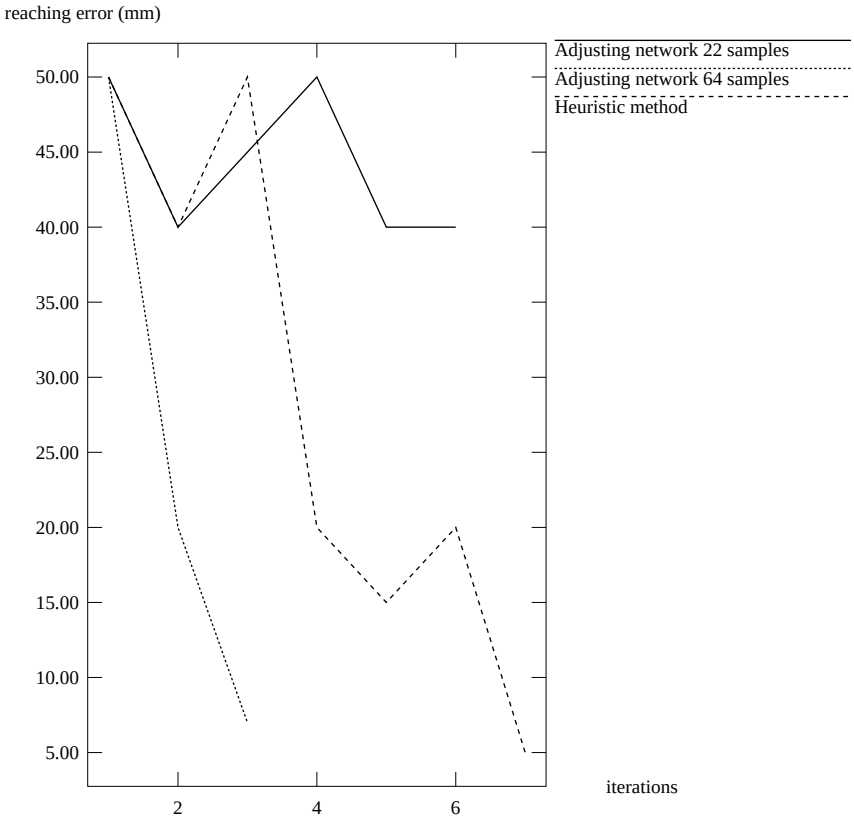

Figure 9: Comparison of heuristic method with adjusting network for gripper position correction. The reaching network training data, consisting of 10 samples for all three graphs, was corrupted to ensure the need for error correction.

5.5 Performance Evaluation

Having tested the various neural network implementations of the controller components, several properties emerge. As the task complexity increases, the size of the neural network needed to learn the mapping also increases. In order to achieve accurate performance, larger training sets are needed, in turn requiring longer training times. For the geometric methods, the more complex tasks require more mathematically intensive implementations and do not possess the ability to adapt on-line to changes in the mapping. Another important observation is that provided reasonable training data, slight architecture modifications to the networks allow very different behaviors to be controlled effectively. Thus, additional behaviors can be added with little effort. The geometric methods, however, all differ considerably from one another. The addition of any new behaviors in a traditional control framework would require a great deal of extra effort for their implementation.

6 NOVICE in operation

The algorithm for co-ordinating the task selection of NOVICE's operations is shown in Figure 10.

Figure 11 shows the results of an experiment where NOVICE used all of its neural controllers to guide itself towards acquiring the target. The entire sequence took approximately seven minutes, with NOVICE moving a total of 4.5 m. Typical times for a sequence not requiring navigation around the table are much shorter, in the order of 30 to 90 seconds. With some simple estimation of the table geometry, the navigation time could be drastically reduced.

7 Adaptability

One of the interesting aspects of neural network controllers is their ability to adapt to changes in the various mappings as new training data is made available. This mechanism reduces the need for off-line recalibration following a perturbation of the system. An important question for our application is how quickly these controllers can adapt and how severe a change in the mapping they can tolerate.

The initial experiments we have performed indicate a surprising robustness to a re-orientation of the cameras. With one camera rotated by roughly 10% (4^0) of its field of view about the y and then x optical axes, NOVICE initially exhibits a slight degradation in performance of all operations. The error for a *centering*

```
while target_obtained = FALSE
        calculate center error
        if center error > CENTER_THRESHOLD
                do_center
        ask reaching network if target sufficiently close to reach
        if target close enough to reach
                do_reach
                calculate reaching error
                if reaching error > REACH_THRESHOLD
                        do_adjust
                else
                        grab target
                        target_obtained ←— TRUE
                        raise arm
        else
                ask approach network if NOVICE too close to table
                if NOVICE too close to table
                        select direction to closer corner of table
                        do_parallel
                        drive past table
                        turn 90^0
                        drive forward
                        turn 90^0 (to face table)
                else
                        do_approach
end while
```

Figure 10: Task selection algorithm used by the master controller.

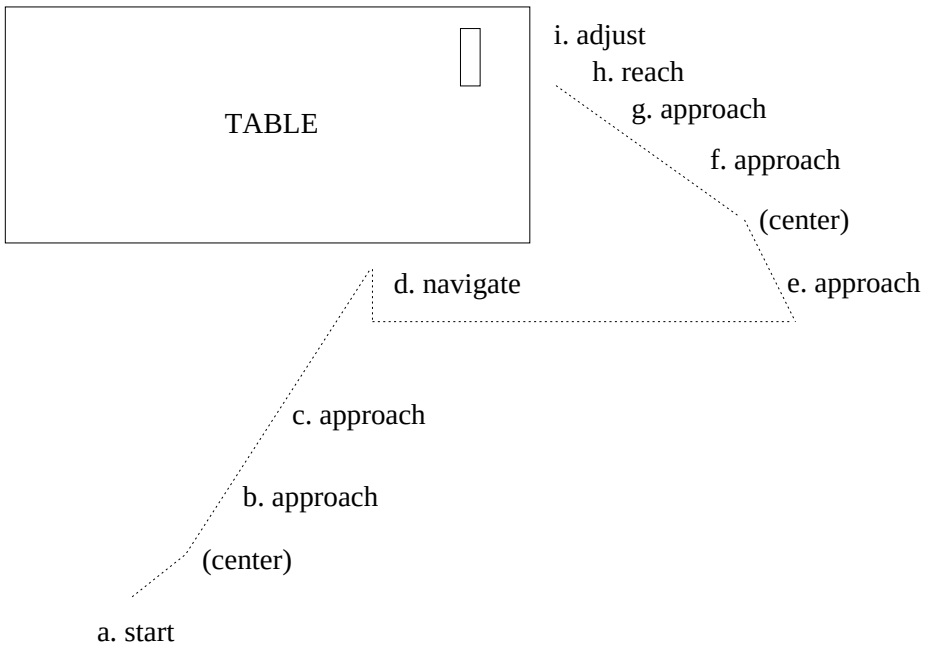

Figure 11: Sample NOVICE path employing all five neural networks.

operation increased slightly, but remained sufficiently low so that no further corrective rotation was required. As a result of the perceived modification of the target positions, T_X and T_Y and their disparity between the two images, the error for a *reaching* operation increased slightly in lateral and vertical displacement as well as depth. The reaching error on NOVICE's first attempt was initially too high to permit grasping. However, as this physical perturbation was similar to the random perturbation of the reaching network training data discussed earlier, an adjustment method was able to correct the reaching position.

Figure 12 compares the control results of the heuristic gripper position, using a reaching network trained with 10 data samples, and an adjusting network trained with 20 data samples, following the camera perturbation. On-line training was disabled for the system, so that the adjusting behaviour of both methods could be compared based solely on their prior mappings. The target location in these experiments was near the center of both images. Note that as the magnitude of corrective movements made by the heuristic method are proportional to the current error, this method outperforms the adjusting network which was trained for making corrective movements no larger than 10 mm.

Another surprising result was the rapid performance improvement observed with on-line training active for all networks following the re-orientation of one camera. After only three docking and reaching sequences following the re-orientation, NOVICE was able to perform the operation successfully with no corrective rotation or reaching control. These results indicate the feasibility of on-line adaptation as an alternative to off-line recalibration.

8 Conclusions

A mobile robot system, controlled by a collection of small neural networks, exhibiting accurate, repeatable behavior of docking and reaching, and which readily adapts to changes in its own configuration, was designed and constructed. We have shown experimentally the feasibility of this approach for performing non-trivial robotic tasks using no additional sensory data besides simplified vision, and demonstrated it to be a useful alternative to traditional methods involving computation of the inverse perspective projection and inverse kinematics.

Our experiments showed that we can combine several primitive modules which perform simple tasks in isolation to obtain complex aggregate behavior as the system reacts to the environment. In addition to

Figure 12: Comparison of reaching error correction with the adjusting network and a heuristic method following left camera rotation of 4^0 about the y axis and right camera rotation of 4^0 about the x axis.

capturing non-linear mappings effectively, we were able to exploit the adaptive properties of neural networks to increase the robustness of our system. The neural controller was seen to be insensitive to slight variations of equipment configuration and tolerant of minor camera perturbations.

NOVICE provides several mechanisms for improving its accuracy. Its neural network controllers constantly adapt themselves to fit whatever data samples they have obtained, thus avoiding the need for off-line recalibration following a slight change of the configuration. In addition, both a heuristic and neural method are available for gripper position adjustment.

Acknowledgements

Thanks are due to Ben Gamsa for his novel solutions to several programming problems, David Wilkes for providing robot simulator code for an early prototype of NOVICE, and Niels daVitoria Lobo for many stimulating discussions regarding computer vision. Thanks also to Piotr Jasiobedzki for his helpful comments, to Professor H. Niemann of the University of Erlangen for pointing out references [10] and [1], and to Dr. Michael Brown of AT&T Bell Labs for suggesting the analysis in the appendix.

Appendix: Derivation of position estimation error

Visual uncertainty in the image planes results in a corresponding physical error of the estimated target position. The magnitude of the physical error can be determined by simple geometry. Referring to Figure 13, we can compute the x and z co-ordinates of the true target position, T, as follows:

The description of line l is

$$z = (x + b/2) \tan \Theta_l \tag{1}$$

The description of line r is

$$z = -(x - b/2) \tan \Theta_r \tag{2}$$

Equating (1) and (2) yields for x

$$x = \frac{b}{2}\left(\frac{\tan \Theta_r - \tan \Theta_l}{\tan \Theta_r + \tan \Theta_l}\right) \tag{3}$$

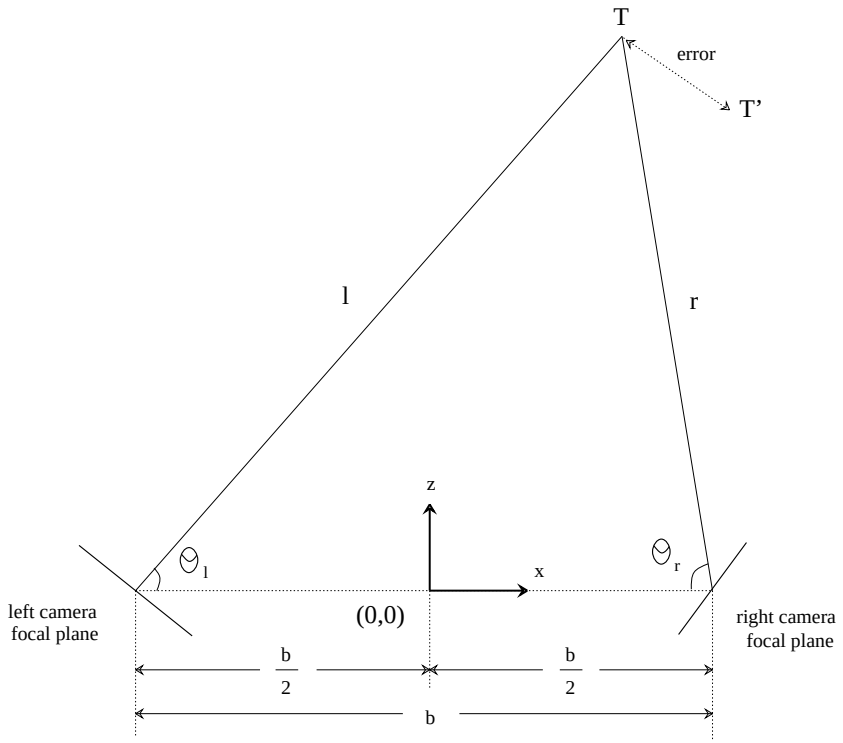

Figure 13: The error between the true target position, T, and an estimated target position, T', due to pixel error in the image planes.

Rewriting (1) and (2) in terms of z results in the following for line l

$$x = z \cot \Theta_l - (b/2) \tag{4}$$

and for line r

$$x = z \cot \Theta_r - (b/2) \tag{5}$$

Equating (4) and (5) yields for z

$$z = \frac{b}{\cot \Theta_r + \cot \Theta_l} \tag{6}$$

Now if a measurement error is made in the camera images, we can find the x and z co-ordinates of the estimated target position, T', by replacing Θ_l and Θ_r with $\Theta_l + \epsilon_l$ and $\Theta_r + \epsilon_r$, where ϵ_l and ϵ_r are the angular measurement errors in the left and right camera images, respectively. We will consider positive values of ϵ_l and ϵ_r to increase the effective angles of Θ_l and Θ_r.

Assuming no error in the y direction, the effect of a measurement error in one or both camera images can

be calculated as the Euclidean distance between T and T' as follows

$$\text{error} = \sqrt{(T_x - T'_x)^2 + (T_z - T'_z)^2} \tag{7}$$

For the Panasonic video cameras set at a 12.5 mm focal length, the field of view encompasses approximately 30^0 horizontally. Therefore, each of the 256 pixel columns represents approximately $30^0/256 = 0.1176^0$.

For sample values of $\Theta_l = \Theta_r = 73.6^0$ and a baseline length of 600 mm, we obtain the following position errors as a result of single pixel errors in one or both images:

image error (pixels)		target position (mm)		position error (mm)
left	right	x	z	
0	0	0.0	1019.3	0.0
1	0	1.1	1023.2	4.0
1	1	0.0	1027.1	7.7
1	-1	2.3	1019.3	2.3
-1	-1	0.0	1011.7	7.6

References

[1] Michael K. Arras, Peter W. Protzel, and Daniel L. Palumbo. Automatic learning rate adjustment for self-supervising autonomous robot control. Technical Report TM–107592, NASA, January 1992.

[2] Jonathan H. Connell. *Minimalist Mobile Robotics: A Colony-style Architecture for an Artificial Creature.* Perspectives in AI. Academic Press Inc., 1990.

[3] Jeremy R. Cooperstock and Evangelos E. Milios. A neural network operated vision-guided mobile robot arm for docking and reaching. Technical Report RBCV TR 92–39, University of Toronto, 1992.

[4] O.D. Faugeras and G. Toscani. Camera calibration for 3d computer vision. In *Proceedings of International Workshop on Machine Vision and Machine Intelligence*, Tokyo, Japan, February 1987.

[5] Jean-Yves Hervé, Rajeev Sharma, and Peter Cucka. Qualitative coordination of a robot hand/eye system. Technical Report CAR–TR–516 or CS–TR–2544, Computer Vision Laboratory, University of Maryland, October 1990.

[6] Geoffrey E. Hinton. Connectionist learning procedures. *Artificial Intelligence*, 40:185–234, 1989.

[7] Michael I. Jordan. An introduction to linear algebra in parallel distributed processing. In David E. Rumelhart and James L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations.* MIT Press, 1986.

[8] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[9] Michael Kuperstein and Jorge Rubinstein. Neural controller for adaptive sensory-motor coordination. *SPIE*, Intelligent Robots and Computer Vision(1002):658–670, 1988.

[10] Thomas M. Martinetz, Helge J. Ritter, and Klaus J. Schulten. Three-dimensional neural net for learning visuomotor coordination of a robot arm. *IEEE Transactions on Neural Networks*, 1(1), March 1990.

[11] R. Paul. *Robot manipulators: mathematics, programming, and control.* MIT Press, 1981.

[12] Dean A. Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3:88–97, 1991.

[13] P. Puget and T. Skordas. An optimal solution for mobile camera calibration. *IEEE International Conference on Robotics and Automation*, 1, 1990.

[14] P. Patrick van der Smagt and Ben J. A. Kröse. A real-time learning neural robot controller. In *Proceedings of International Conference on Artificial Neural Networks, Espoo, Finland*, pages 351–356, 1991.

[15] B. Widrow and M.E. Hoff. Adaptive switching circuits. *Institute of Radio Engineers, Western Electronic Show and Convention, Convention Record, Part 4*, pages 96–104, 1960.