We analyze a neural network implementation for puck state prediction in robotic air hockey. Unlike previous prediction schemes which used simple dynamic models and continuously updated an intercept state estimate, the neural network predictor uses a complex function, computed with data acquired from various puck trajectories, and makes a single, timely estimate of the final intercept state. Theoretically, the network can account for the complete dynamics of the table if all important state parameters are included as inputs, an accurate data training set of trajectories is used, and the network has an adequate number of internal nodes. To develop our neural networks, we acquired data from 1500 no-bounce and 1500 one-bounce puck trajectories, noting only translational state information. Analysis showed that performance of neural
networks designed to predict the results of no-bounce trajectories was better than the performance of neural networks designed for one-bounce trajectories. Since our neural network input parameters did not include rotational puck estimates and recent work shows the importance of spin in impact analysis, we infer that adding a spin input to the neural network will increase the effectiveness of state estimates for the one-bounce case.

1. INTRODUCTION

To carry out a study of issues inherent in hybrid, intelligent control of robotic systems, we have developed a robotic air hockey system. Basic to the air hockey problem is control of a sliding cylindrical puck on a bounded, rectangular, low-friction surface. This puck is controlled through impacts with a cylindrical paddle, which is restricted to a specific area on the surface and maneuvered with a robotic manipulator. Provided estimates of a puck’s state at a selected intercept time and proper impact models, the manipulator can maneuver the paddle to strike the puck on a desired trajectory. The robotic system must predict the intercept state of the puck long before the strike to effectively prepare for the strike. A reliable puck state estimator must be designed to provide accurate future state estimates from current state estimates. For the air hockey system to be successful, the state estimator must provide timely, accurate puck estimates for the manipulator to be in position to react to the puck’s future state.

Unlike original prediction techniques using least squares and impact modeling, we consider puck prediction using neural networks. Original prediction techniques were based on dynamic assumptions that did not account for the physical unpredictability of the table surface. A neural network provides a new modeling approach that uses a large number of trials to computationally select a dynamic model rather than using physics to assume one. The final dynamic model chosen is the one that minimizes the difference between the outcome of the known trials and the outcome the model predicts. If the trial data are complete and the inputs to the network provide all the necessary dynamic variables, the neural network model will be superior to the original dynamic model because it will account for the unpredictability of the system along with the basic physical assumptions.

1.1. System Architecture

To provide a more complete understanding of the application involved, we shall describe the architecture of the air hockey system. Figure 1 provides a visual overview of the system.

The robot’s arm is driven by three Yokogawa direct-drive motors. These motors receive torque signals and send encoder feedback through a VME Galil DMC-530 board located in a nearby VME cage. This Galil board is commanded by a TMS-320 DSP on a VME Pentek 4283 board. Software run on this 4283 board is downloaded from a Sun Sparc 10 via a Bit3 SBUS to VME interface. The air hockey code is compiled on this Sparc 10 using the Pentek Swiftools suite. The vision system is implemented separately on the VME cage.

A camera above the air hockey table sends frames to a Max Video 20 Datacube board. The board is controlled by Imageflow software running on the Sparc 10. The board is commanded using a Performance Technologies SBUS to VME card, the second SBUS to VME card in the cage. The vision software first obtains an empty table frame and subtracts this from preceding frames. When the puck is detected, an algorithm is begun that will provide puck windowing. In this way, the system can search for a new centroid around the area where we last saw the puck. Interferences, like a hand waving across the vision plane or noise from inadequate lighting, can be ignored. The Datacube leaves...
puck centroid information available on the bus. The TMS-320 polls the Datacube during software execution.

The DSP control code consists of a continuous loop which itself performs a wide variety of tasks. The current puck centroid information is read. Based on this information and past centroids, a puck state estimator predicts the probable interception point. In this article, the state prediction is done with a neural network. Based on the prediction provided, the software calculates the best angle of attack and swing time. Straight goal shots or one-bounce trajectories are calculated. Nonlinear control techniques derive the necessary motor torques to achieve these trajectories. These torques are sent to the motor and the cycle then repeats.

1.2. Outline

Section 2 will begin with a description and analysis of the original puck state estimation techniques and will continue by presenting the neural network estimation scheme. In section 3, we provide results from the application of the new estimation technique. Within section 4, we present the conclusions of this article.

2. STATE ESTIMATION

In this section, we will discuss traditional techniques used for state estimation and then provide a replacement with neural networks.

2.1. Conventional Estimation Techniques

For the air hockey system, a number of physical models could be used. The most simplistic model considers the puck to be an extended particle, all collisions to be nearly elastic, and forces tangent to impact surfaces negligible (thus the puck’s rotation is unimportant). The most complicated model available includes dynamic effects of spin, friction, inelastic collisions, and fluid dynamics (for the airflow). For the most complicated models, work can be referred to on the modeling of ice hockey pucks as well as on the study of the dynamics of impulsive manipulation. Both of the works are computationally intensive and include some dynamics that do not apply to the air hockey problem.

There are many problems that occur when using the most complicated models. To model the exact dynamics of the table would require an understanding of the complete topology of the table along with the material properties of the components involved. We find it extremely difficult to physically model all aspects of the environment. Another problematic aspect of modeling the air hockey system is the possibility that the dynamics may change over time. The air supply for the table may vary in pressure. The table may be scuffed by the puck or paddles, which could result in dynamic changes to the system. To attempt to correct for these changes outside of the defined model parameters is a difficult problem.

Because of these difficulties, original prediction models used the most simplistic impact model and assumed linear translation of the puck. This model assumed a constant coefficient of restitution for impacts between puck and the paddle as well as puck and table edges. Friction and spin had been ignored until recently. These previous models provided a general overview of the dynamics, but they were not exact.

We desired to find a new prediction strategy that would completely model the dynamics of the system, yet was feasible to obtain. In the course of our study we examined a number of works. Prediction theory has a broad base in the literature. Many techniques exist for performing filtering of signals to compensate for noise and disturbances in a control theoretic frame work. However, in the case of simple linear motion, most of these techniques become little more than weighted averaging methods. None of these methods would provide much assistance.

2.2. Neural Network-Based States Estimation

Research in the area of neural networks showed that prediction could be performed which accounted for the entire dynamics of the air table and could be updated as the system properties changed. We consider a straightforward prediction technique using neural networks.

Neural networks are trained functions. One initially selects the input and output relationship. The functions are modeled from experimental training data relating the input to the output. The function is a representation of the trials.

In the case of our state prediction scheme, we define the input parameters as the state of the puck at the opponent end of the table and our output as the state of the puck at a desired intercept area of the table. We examined the parameters that would be important to include in the input and output.
Figure 2. Measured puck parameters: opponent line—the line in which the puck crosses as it begins motion down the air table. As the puck crosses this line, the current puck state is measured; intercept line—the line where the robot’s end-effector strikes the puck and also the line for which the neural network predicts the future state of the puck from the current state measured at the opponent line; passing location \((x_m[m])\)—the position at which the puck crosses the opponent line; incoming angle \((\theta_{in}[\text{rad}])\)—the angle at which the puck crosses the opponent line; incoming velocity \((v_{in}[\text{m/sec}])\)—the velocity of the puck as it crosses the opponent line; intercept location \((x_{out}[m])\)—the position at which the puck crosses the intercept line; outgoing angle \((\theta_{out}[\text{rad}])\)—the angle at which the puck crosses the intercept line; outgoing velocity \((v_{out}[\text{m/sec}])\)—the velocity of the puck as it crosses the intercept line; travel time \((\Delta t[\text{sec}])\)—the time required for the puck to travel from the opponent line to the intercept line.

For the remainder of this article we will note the state of the puck at the opponent line as the input state and the state of the puck at the intercept line as the output state. The input state includes passing location, incoming angle, and incoming velocity of the puck. The output state includes the intercept location, the outgoing angle, the outgoing velocity, and the travel time. We use a neural network to provide a predicted output state given the input state.

We introduce the multilayer back propagation neural network, which has inputs, hidden layers, and outputs. The back propagation learning algorithm is one of the most important historical developments in neural networks. If a multilayer neural network has enough hidden layers, it is able to approximate any nonlinear function.

Figure 3 shows the implementation of the neural network and should be used as a reference for the rest of this section. We choose the passing location \((x_m)\), incoming angle \((\theta_{in})\), and incoming velocity \((v_{in})\) on the opponent line as input neurons while we select only one neuron as an output neuron. For each required output, we use a separate neural network. The outgoing angle \((\theta_{out})\), outgoing velocity \((v_{out})\), intercept location \((x_{out})\), and the travel time \((\Delta t)\) are used as outputs. For each neural network, we have a single hidden layer which has 6 or 7 neurons depending on the network. The learning speed of the back propagation neural network is affected by several factors, including the initial weights, initial bias, learning rate coefficient, the number of neurons in the hidden layer, etc. Furthermore, the highly nonlinear characteristics of the function to be learned increases the learning time required. To reduce the learning time, separate neural networks for each output rather than a single fully interconnected network are used. We used the bipolar sigmoid function as the activation function and used a momentum term in the training algorithm to enhance the convergence speed of the neural network. Initial weights are generated randomly. The learning rate is initialized as a value from 0 to 1.

The \(q\)th hidden layer produces an output of

\[
z_q = a(\text{net}_q) = a \left( \sum_{j=1}^{m} w_{qj} x_j \right)
\]

Figure 3. Multilayer neural network to predict intercept states.
where $a()$, $v_{iq}$, and $x_i$ represent the bipolar sigmoid function, the weights between inputs and the hidden layer, and the inputs of neural network, respectively. $m$ is the number of neurons of input layer. The $i$th output of the output layer is given by

$$y_i = a(\text{net}_i) = \sum_{q=1}^{l} w_{iq} z_q$$

(2)

where $w_{iq}$ is the weight for each neuron in the hidden layer and the output layer and $l$ is the number of neurons in the hidden layer. The above equations indicate the forward propagation of input signals through the layers of neurons. Next, we shall consider the error signals and their back propagation. We first define a cost function as in the following equation.

$$E(w) = \frac{1}{2} \sum_{i=1}^{n} (d_i - y_i)^2$$

(3)

where $d_i$ is the $i$th reference output of neural network and $y_i$ represents the $i$th real output of the neural network where $i$ is 1. According to the gradient-descent method, the weights in hidden-to-output connections are updated by

$$\Delta w_{iq} = -\eta \frac{\partial E}{\partial w_{iq}} = \eta \cdot \delta_{iq} \cdot z_q$$

(4)

where

$$\delta_{iq} = [d_i - y_i] \cdot [a'(\text{net}_q)]$$

is the learning rate.

For the weight updates on the input-to-hidden layer connections, we use the chain rule with a gradient-descent method and obtain the weight update on the link weight connection $j$th in the input layer to $q$th in the hidden layer.

$$\Delta v_{iq} = -\eta \frac{\partial E}{\partial v_{iq}} = \eta \cdot \delta_{iq} \cdot x_j$$

(5)

To make the neural network learn, we needed trial data to train it. Sample data were obtained from 1500 no-bounce puck trajectories and 1500 one-bounce trajectories. Table I shows examples of data measured.

### 3. ANALYSIS OF THE NEURAL NETWORKS AND DATA

#### 3.1. Analysis of the Neural Networks

Let us consider the neural network described in section 2 for estimation of the intercept puck state. We created networks using data from 1500 no-bounce, 1500 one-bounce, and 1500 mixed-bounce puck trajectories. The mixed-bounce data combined data from 750 no-bounce and 750 one-bounce trajectories. Since an air hockey system has a complicated nonlinear characteristic due to the surface friction and the puck spin, it is difficult to generate the exact dynamic model for it on a computer. So the immediate trials to obtain the model were performed using the actual air hockey system rather than computer simulation. The puck was hit by the paddle with various speeds and various spins in different positions. The velocity, outgoing angle, intercept position, and traveled time of the puck

<table>
<thead>
<tr>
<th>Passing location (m)</th>
<th>Incoming velocity (m/s)</th>
<th>Incoming angle (rad)</th>
<th>Intercept location (m)</th>
<th>Outgoing velocity (m/s)</th>
<th>Outgoing angle (rad)</th>
<th>Travel time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1440</td>
<td>2.4958</td>
<td>-0.0056</td>
<td>0.1054</td>
<td>2.3273</td>
<td>0.0309</td>
<td>0.3800</td>
</tr>
<tr>
<td>-0.0009</td>
<td>1.5911</td>
<td>0.1006</td>
<td>-0.0985</td>
<td>1.4792</td>
<td>0.1074</td>
<td>0.6100</td>
</tr>
<tr>
<td>0.1367</td>
<td>1.7823</td>
<td>-0.0109</td>
<td>0.1118</td>
<td>1.7042</td>
<td>0.0537</td>
<td>0.5450</td>
</tr>
<tr>
<td>-0.2717</td>
<td>2.1518</td>
<td>0.0278</td>
<td>-0.2824</td>
<td>2.1628</td>
<td>-0.0048</td>
<td>0.4450</td>
</tr>
<tr>
<td>-0.3508</td>
<td>2.4654</td>
<td>0.0778</td>
<td>-0.4121</td>
<td>2.2599</td>
<td>0.0396</td>
<td>0.4100</td>
</tr>
</tbody>
</table>
were then calculated from data obtained by the camera located above the air hockey table. We used the multilayer network discussed in section 2. The passing location neural network was trained for 1000 times for each training set and the remaining neural networks were trained 500 times for each training set.

Figures 4 through 7 show the transition of mean square error between the actual output and the neural network predicted output during learning for intercept location, outgoing velocity, outgoing angle, and travel time, respectively. Mean square error is defined as

\[
MSE = \sum_{k=1}^{N} (y_{ref_k} - y_{out_k})^2
\]

Figure 4. Mean square error for the intercept location ($x_{\text{out}}$).

where MSE is mean square error and $N$ is the number of observed samples. Each figure has three mean square error lines representing no-bounce data, one-bounce data, and mixed-bounce data. Mean square errors for all cases decrease with respect to learning iterations. The final mean square error after learning for the no-bounce data is less than that for one-bounce data in all cases. Similarly, the mean square error for one-bounce data after learning is less than that for mixed data for all cases except for intercept location.

The neural networks were trained, we compared estimated future intercept states of the puck with the actual final states. The results are shown in Figures 8 through 11. Figure 8 shows the prediction results for intercept location of the puck.

Figure 5. Mean square error for the outgoing velocity ($v_{\text{out}}$).

Figure 6. Mean square error for the outgoing angle ($\theta_{\text{out}}$).

Figure 7. Mean square error for the travel time ($\Delta t$).
The no-bounce case provided much better prediction than any other case. Figure 9 represents the prediction results for outgoing velocity. In this case, the performances for one-bounce trials and no-bounce trials are nearly identical [Fig. 9(a) and (b)]. In all cases, as the velocity of puck increases, we find more prediction error. This may be caused by measurement error associated with the constant sampling rate of the vision system. Figure 10 shows the results for incoming angle prediction of the
puck on the intercept line. Again, the no-bounce case has the best performance. Figure 11 presents the travel time prediction results. The performances for the one-bounce case and the no-bounce case are nearly identical. Much like the outgoing velocity estimate, as travel time increases, the prediction is less accurate. For large travel times, the trajectory of the puck is affected by the nonplanar properties on the table, and therefore, large travel times are difficult to predict.
To analyze the neural network results further, we used the following equations.

\[
M = \frac{\sum_{k=1}^{N} X_k}{N}
\]

(7)

\[
SD = \frac{\sum_{k=1}^{N} |X_k - M|}{N}
\]

(8)

where \( M \) and \( SD \) are mean and standard deviation, respectively. \( N \) is the number of observed samples. We present the standard deviation and mean square error for all output predictions. We summarize the results in Table II. Using the standard deviation, we can estimate the predicted error of all cases statistically.

3.2. Analysis of Training Data

From section 3.1, we find that neural network prediction performance for no-bounce shots was better than performance for one-bounce shots. To discover why performance was better in the no-bounce case, we analyzed the data used to train the networks.

We first considered the characteristics of the no-bounce data. Because friction or gravitational effects on the puck could vary according to location on the air table, we investigated the linearity of the puck trajectory for the 1500 no-bounce samples. We calculated the mean and standard deviation of the difference between \( \theta_{in} \) and \( \theta_{out} \) using Eqs. (7) and (8). The mean was 0.0053 rad and standard deviation was 0.0307 rad. From these results, we concluded that the trajectory of puck was nearly straight.

In the one-bounce case, we add a single impact but otherwise have a trajectory no different from the no-bounce trajectory. We studied the effect of this impact on the data. In studying the impact, we defined the rebound coefficient for the one-bounce case as the following equation:

\[
\alpha = \frac{\tan(\theta_{out})}{\tan(\theta_{in})}
\]

(9)

We calculated the mean and standard deviation of the rebound coefficient for the 1500 one-bounce trials and present our results in Table III. The mean was 0.8810 and standard deviation was 0.1026. By the empirical rule of statistics, approximately 98%
of the observations will lie within ±2 SD of the mean of the rebound coefficient α. Hence, 98% of the rebound coefficient observed had values between 0.6758 and 1.0862. The rebound coefficient varied widely.

From Eq. (9), we can estimate the varying range of θout in the case of a one-bounce trial using the following equation:

\[
\theta_{\text{out}} = \tan^{-1}\left( (\alpha \pm \delta \text{SD}) \cdot \tan(\theta_{\text{in}}) \right)
\]  

(10)

If n equals 2 and θin is 45°, then 98% of θout lie between 34.0508° and 47.3661°, that is, the variation range of θout is 13.3153°.

This large deviation may be caused by a varying restitution coefficient (defined by the material properties of the objects colliding) or puck spin. Previous work has shown that the restitution coefficient between the puck and the air table is nearly constant regardless of collision location and also that spin and friction are important factors for two-dimensional collisions. Hence, we can assume that the variation of the rebound coefficient is caused by friction and puck spin. Our neural networks did not apply a spin input. In the future, we will find it necessary to add puck spin as an input component for more accurate predictions.

4. CONCLUSIONS

We have successfully applied a puck state prediction scheme using neural networks. The inputs and outputs of these networks were the translational state components of the puck. We have found that the prediction of networks designed for no-bounce trajectories provided better prediction than prediction for one-bounce or mixed-bounce trajectories.

We then analyzed the difference in one-bounce and no-bounce prediction. We investigated the variation of the rebound coefficient for one-bounce cases and found the standard deviation to be large. Because only translational state components were measured, we assume that estimation for the one-bounce case will improve if spin was measured and added as a necessary input.

In the future, we wish to expand the capabilities of the networks to account for spin. To do this, we must develop better vision techniques to estimate spin at high puck velocities. Vision work is currently underway to investigate this restriction.

REFERENCES