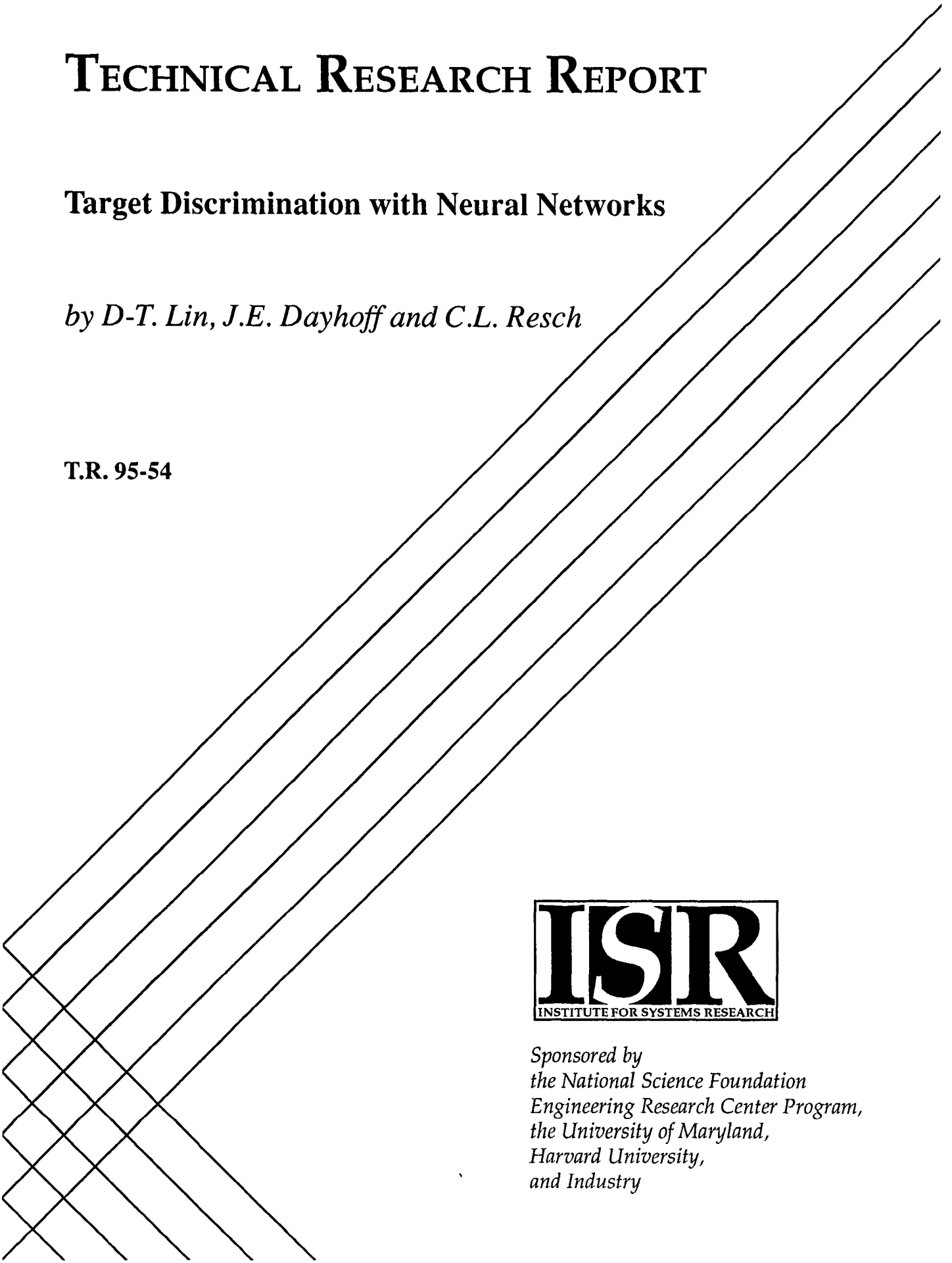
TECHNICAL RESEARCH REPORT

Target Discrimination with Neural Networks

by D-T. Lin, J.E. Dayhoff and C.L. Resch

T.R. 95-54



Sponsored by
the National Science Foundation
Engineering Research Center Program,
the University of Maryland,
Harvard University,
and Industry

Target Discrimination with Neural Networks*

Daw-Tung Lin†, Judith E. Dayhoff† and Cheryl L. Resch‡

†Institute for Systems Research

University of Maryland

College Park, MD 20742

‡ Applied Physics Laboratory

Johns Hopkins University

Laurel, MD 20723

January 16, 1995

*This work was supported by the Applied Physics Laboratory of Johns Hopkins University and the Institute for Systems Research at the University of Maryland (NSF CDR-88-03012).

Abstract

The feasibility of distinguishing multiple type components of exo-atmospheric targets is demonstrated by applying the Time Delay Neural Network ($TDNN$) and the Adaptive Time-Delay Neural Network ($ATNN$). Exo-atmospheric targets are especially difficult to distinguish using currently available techniques because all target parts follow the same spatial trajectory. Thus classification must be based on light sensors that record signal over time. Results have demonstrated that the trained neural networks were able to successfully identify warheads from other missile parts on a variety of simulated scenarios, including differing angles and tumbling. The network with adaptive time delays (the $ATNN$) performs highly complex mapping on a limited set of training data and achieves better generalization to overall trends of situations compared to the $TDNN$, which includes time delays but adapts only its weights. The $ATNN$ was trained on additive noisy data and it is shown that the $ATNN$ possesses robustness to environment variations.

1 Automatic Target Recognition

Automatic Target Recognition (ATR) is a highly challenging problem. It involves extraction and discrimination of critical information from complex and uncertain data. Due to target signature variability, environmental changes, and a limited database, the traditional approaches of signal processing and rule-based expert systems have been only partially successful [13]. Neural net technology offers a number of tools such as learning and adaptation, generalization and robustness, feature extraction and hardware implementation, which could form the basis of a fruitful approach to the ATR problem [4, 13, 1]. Neural net architectures

with dynamic and temporal capabilities are more promising for signal analysis.

The time-delay neural network (*TDNN*) proposed by Waibel et al [15] employs time-delays on connections and has been successfully applied to phoneme recognition [16, 17, 14]. The time-delay neural network also classifies spatiotemporal patterns and provides robustness to handling noise and allowing graceful degradation [9]. In the *TDNN* architecture, each neuron takes into account not only the current information from its input neurons of the previous layer, but also a certain amount of past information from those neurons due to delays on interconnections. Typically the time delays are evenly spaced over a time interval called the frame window, although arbitrary time delays may be used. Training is done with spatiotemporal patterns, and the classification of those patterns is reported at each time step by the output layer. After training, weights are strengthened along interconnections whose time delays are important to recognition.

The Adaptive Time Delay Neural Network (*ATNN*), which adapts time delays as well as weights during training, is a more advanced version of the $TDNN$. The result is a dynamic learning technique for spatiotemporal classification [7]. We use an algorithm with adaptive time-delays [8, 3] which is a powerful tool for dynamic learning. The *ATNN* model employs modifiable time delays along the interconnections between two processing units, and both time delays and weights are adjusted according to system dynamics in an attempt to achieve the desired optimization. The adaptation of the delays and weights are derived based on the gradient descent method to minimize the energy or cost function during training. Weight modification is based on error back-propagation ([10]) and the mathematical derivation of the time-delay modifications is done with a gradient descent approach [2, 3, 8, 7]. The weights and time-delays are updated step by step proportional to the opposite direction of

the error gradient respectively. Processing units do not receive data through a fixed time window, but gather important information from various time delays which are adapted via the learning procedure. With these mechanisms, the network implements the dynamic delays along the interconnections of the *ATNN*.

We report here the application of the $ATNN$ to exo-atmospheric target discrimination. The scenario of this problem is schematically illustrated in Figure 1. The components of a vehicle (a threat) will be separated and detected by a visual sensor after it is launched. The visual sensor records intensity and size over time for each component. Among these components, we are only interested in the warhead component as the other pieces are non-warhead pieces. The objective is to discriminate between these components regardless of different aspect angles, tumbling or broken pieces and find the target component.

This is a very important and yet difficult task in target component discrimination, as it states in APL technical report by Resch [11], "The ability of a kinetic kill vehicle (KKV) to discriminate between warhead and booster parts or decoys is critical to theater ballistic target defense (TBMD). Exo-atmospheric targets are especially difficult to distinguish using currently available techniques because all target parts follow the same trajectory during the exo-atmospheric portion of the flight." Furthermore, only one sensor is assumed to be available.

The sensors used in this analysis have dedicated specifications. The maximum radiance and size versus time are obtained from these sensors for all four components (denoted in this context as warhead, oxidizer tank, fuel tank, and fins, in which warhead is the target we are interested in). Furthermore, these components may tumble with different aspect angles or break into several pieces. One of the many things that makes ATR so hard is that the same

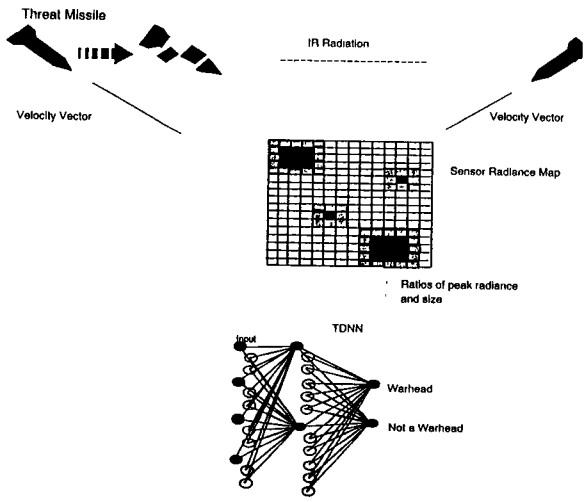


Figure 1: Target components discrimination.

target can vary wildly in appearance depending on aspect angles, atmospheric effects, and other variables.

We address this problem by using both the $TDNN$ and $ATNN$ and conclude that the $TDNN$ has good performance, and the $ATNN$ can make further improvements in this performance.

2 TDNN Technique

2.1 Simulation Setups

The $TDNN$ was first used to perform the discrimination task of distinguishing the warhead from the other missile parts. Data was supplied by C Resch, from APL simulation, labeled "threat A".

We used a $TDNN$ with an input layer with six inputs, one hidden layer with three hidden units and an output layer with two units. The number of time-delay on the first layer is

four and on the second layer is six ($6 \xrightarrow{4} 3 \xrightarrow{6} 2$). Time-delays on each pair of connections between neurons are fixed and are consecutively and equally spaced as $0, \tau, 2\tau, ...$, where τ is the data sampling interval. Data is input to the network sequentially starting before intercept and ending very shortly before intercept. The target value is one or zero representing either true or false targets for each particular training set composed of the ratios described above. The following techniques were used on this data discrimination problem:

1. A symmetric sigmoid function was used on all processing units. This appears to be advantageous in convergence and recognition performance.

2. The data scaling procedure was revised so that the same scaling factor F_s was used on all data. The data was scaled and offset to be symmetric at 0 with range [-0.5:0.5]. The scaling factor F_s was calculated from the training set by the following equation (Equation 1:

$$F_s = max(\mathcal{S}_{train}) - min(\mathcal{S}_{train}) \tag{1}$$

where $\mathcal{S}_{train}$ stands for the training data space.

The training data is scaled and offset to be symmetric to 0 by Equation 2:

$$network\ input = \frac{(\mathcal{S}_{train} - p_{mid})}{F_s} \tag{2}$$

where $p_{mid} = min(\mathcal{S}_{train}) + F_s/2$.

data set	Schedule 2	Schedule 1
training set	100%	89.61%
test set#1	100%	89.14%
test set#2	59.4%	53.18%
test set#3	59.4%	50.00%
test set#4	93.4%	81.82%
test set#5	97.9%	96.59%

Table 1: Successful identification rate of different test sets by using $TDNN$ trained on 0.5 second increment data.

Overall	Schedule 2	Schedule 1
iterations	5565	14323
RMSE	0.044	0.105

Table 2: Over all performance of $TDNN$ trained on 0.5 second increment data.

2.2 Simulation Scheduling and Analysis

Two training schedules were used and tested on five different test sets. Each test set contains more than 24 trajectories in time. These schedules are described as below:

Schedule 1: This training schedule was to begin adapting weights after the first segment of data entered the network [11]. However, we did not allow data from a previous run to remain in the network at the same time that data from a new run entered the network.

Schedule 2: A new training schedule was used in which data filled the network before weight adaptation began. Performance improved with this training schedule as shown in Table 1. Fewer iterations of training were required (only 5565 as compared with 14,323 shown in Table 2).

For both training schedules, there was considerably less possibility of confusion of components fuel tank or fins with target as compared with the previous study [11]. Usually

the fuel tank and fins were clearly identified as not being the target very early. This is an improvement on previous results.

We also used various numbers of hidden units to see if performance would improve. The $TDNN$ was trained to perform target discrimination with the new scaling procedure and with training by Schedule 2, described above. Table 3 summarizes its performance for three hidden units ($3h'$) and five hidden units ($5h'$), as compared to our previous results labeled $3h$ (with three hidden units). Performance was better for the third, fourth, and fifth data sets. The results of each test data set with the trained $TDNN$ with three hidden units are shown in Appendix A Figure 1 to Figure 24.

Another experiment was performed to include additional data in the training set, with different aspect angles for different components. Previous training experiments were limited to data with the same aspect angle for different components. Performance improved on the third, fourth, and fifth data sets compared to the benchmark $3h$ run, but only the fifth data set showed an improvement over other runs with the revised scaling procedure. Performance for this experiment is labeled $3h''$ in Table 3. As a whole, the network identified out the target correctly in all cases. The results show that the smaller the aspect angle on the warhead, the more difficulty the network had in distinguishing the warhead as the target.

2.3 Improvements with Higher Sampling Rate Data

We found that improvements could be made in performance using data with a higher sampling rate. We tested whether faster time sampling improved recognition of target versus non-target parts for the more difficult data where different parts were at different aspect

data set	$3h'$	$5h'$	$3h$	$3h''$
#1	89.61%	89.61%	89.61%	85.61%
#2	89.14%	88.89%	89.14%	88.13%
#3	66.36%	65.00%	53.18%	64.09%
#4	72.72%	70.45%	50.00%	72.73%
#5	82.72%	79.55%	81.82%	83.64%
#6	96.59%	94.89%	96.59%	89.77%

Table 3: Comparison of discrimination performance with different data sets, various training methods and different network topologies.

angles and some parts were broken into pieces as described in Table 4.

The data considered were from two different time-sampling rates: every half second and every tenth second. When using tenth second data, the time delays at the first layer were initially set to (0.0, 0.1, 0.2, 0.3) seconds, and with the half second data they were initially set to (0.0, 0.5, 1.0, 1.5) seconds. Time delays were evenly spaced on the second layer similarly.

We obtained 100 % performance on the faster sampling data. Identification occurred much faster for the tenth second sampling than on the previous data set with half second sampling. In all cases, with the tenth second sample data, the activation became high rapidly for the target and low rapidly for the non-target pieces.

For comparison, Figures 2 to 9 show some of the simulation results on the same scenarios but with two different sampling intervals. On the left hand side are the results from the 0.1 second sampling interval (Figures 2, 4, 6, 8). On the right hand side are the results of the 0.5 second interval data (Figures 3, 5, 7, 9). More results of different aspect angles, tumbling angles and broken pieces are shown in Appendix B Figures 1 to 8.

All figures from tenth second interval data show quick and correct identification. Several of the scenarios with half second interval data had some confusion, at least at first, in the

Trajectory Aspect Angle	Aspect Angle				Broken Pieces			
	wh	ot	ft	fins	wh	ot	ft	fins
30°								
60°								
90°								
90°	90°	60°	60°	30°				
90°	60°	90°	90°	30°				
90°	30°	60°	90°	30°				
90°						front 1/3		
30°						front 1/3		
30°						back 2/3		front 1/3 c2 *
90°						front 1/3		middle 1/3 c2 *
90°						front 1/3	front 1/3	middle 1/3 c2 *
30°						front 1/3		middle 1/3 c2 *
90°	time 323 - 325							

Note: *: fins are replace with other broken part

Table 4: Variation of 0.1 second increments data

identity of the target parts on half second signal data. But with the tenth second data, this confusion was eliminated. These cases are shown by comparing Figure 2 to 3, Figure 4 to 5, Figure 6 to 7, and Figure 8 to 9. Thus the new network that depends on tenth second time sampling overcomes limitations in the previous network and data. We conclude that in the application of the time-delay neural network, time sampling of every tenth second significantly improves performance in terms of speed of identification and percentage of correct recognition.

2.4 Number of Time Steps versus Sampling Rate

In the previous section, we showed that data sampled at tenth second intervals gave superior performance to data sampled at half second intervals. We have explored the reason for this difference, and tested the hypothesis that the total number of samples influences the

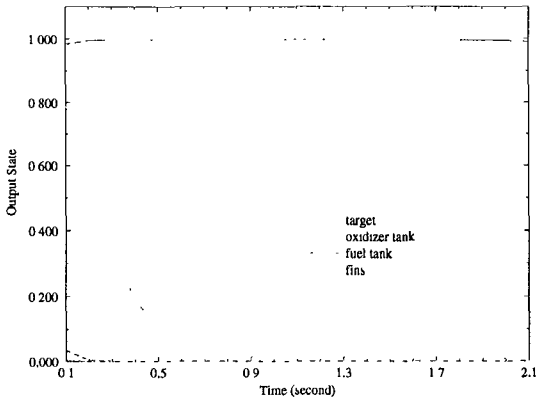


Figure 2: Recognizing target with aspect angle 30°, sampling time interval 0.1 sec.

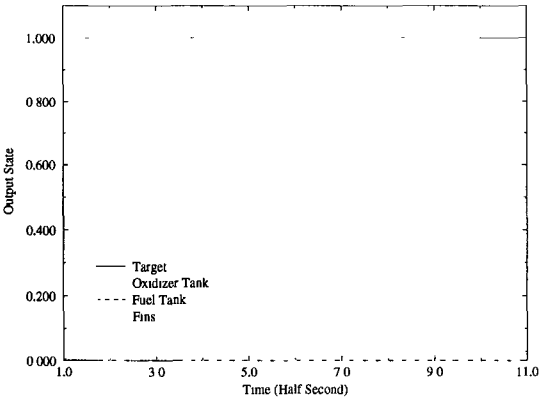


Figure 3: Recognizing target with aspect angle 30°, sampling time interval 0.5 sec.

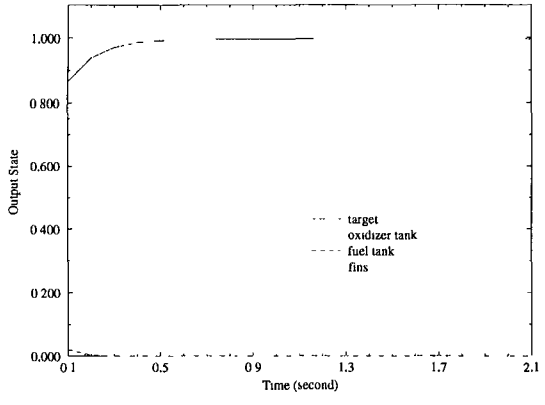


Figure 4: Recognizing target with aspect angle 60°, sampling time interval 0.1 sec.

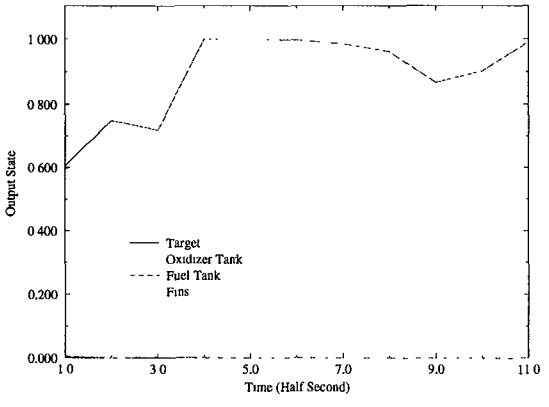


Figure 5: Recognizing target with aspect angle 60°, sampling time interval 0.5 sec.

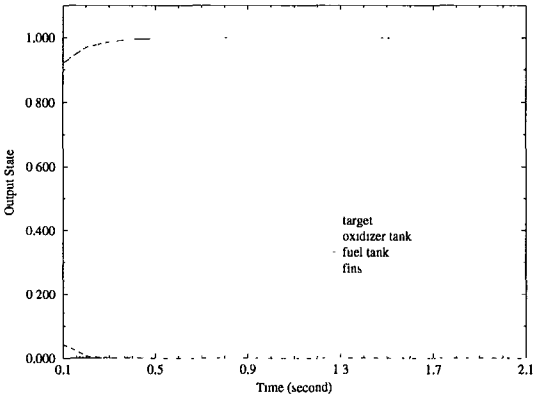


Figure 6: Recognizing target with aspect angle 90°, components 1 to 4 tumble at 90°, 60°, 60°, and 30° respectively, sampling time interval 0.1 sec.

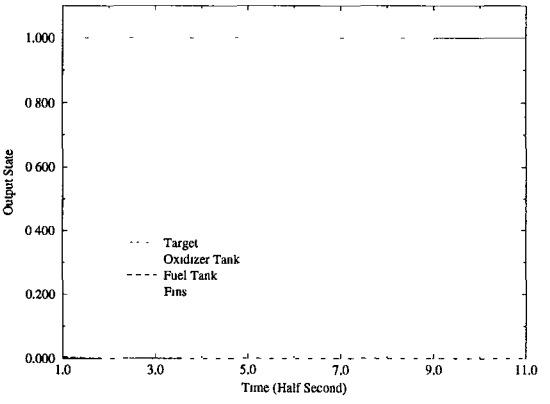


Figure 7: Recognizing target with aspect angle 90°, components 1 to 4 tumble at 90°, 60°, 60°, and 30° respectively, sampling time interval 0.5 sec.

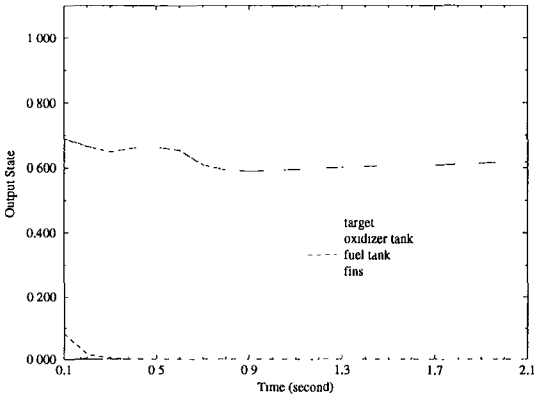


Figure 8: Recognizing target with aspect angle 90°, components 1 to 4 tumble at 30°, 60°, 90°, and 30° respectively, sampling time interval 0.1 sec.

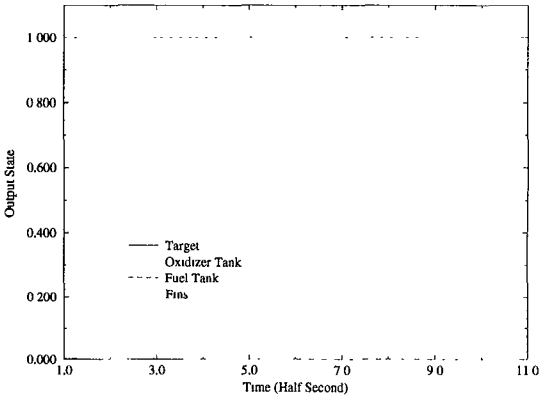


Figure 9: Recognizing target with aspect angle 90°, components 1 to 4 tumble at 30°, 60°, 90°, and 30° respectively, sampling time interval 0.5 sec.

performance. There were twice as many samples in the tenth second data sets as in the half second data sets (the duration of the tenth second data sets is two seconds, while the duration of the half second data is five seconds).

We trained the network on only half of the time samples from the tenth second data. The simulation results in each situation are shown in Appendix C Figures 1 to 12. The resulting performance was still 100%, and the tenth second data still performed in a superior fashion compared to the half second data.

The conclusion is that the number of time samples is **not** the cause of superior performance in the tenth second data. The superior performance results from the higher sampling rate. The radiance data used for the analysis of 0.1 second increments is much smoother than that used in the 0.5 second increment data. Furthermore, after seven seconds until intercept, the pieces begin to take up more than one pixel of the 512 by 512 grid sensors. This causes a change in slope in the data when changing from one pixel to two pixels. During the 0.1 second data, these changes do not occur. The $TDNN$ in this analysis performs well possibly due to the data being smoother and the sensor staying locked on one pixel. A more realistic two pixel data will be analyzed in the next section.

3 ATNN: Enhanced Performance

3.1 Earlier Discrimination on More Realistic Sensor Data

In Section 2.3, we have shown that data sampled at tenth second intervals gave superior performance to data sampled at half second intervals with the $TDNN$. We obtained sim-

ulated data from C Resch at APL for a duration of four seconds. We applied the $ATNN$ to similar data, sampled at tenth second intervals. Furthermore, we needed a longer data scenario to allow more possibilities for time-delays internal to the network. We applied both the $ATNN$ and $TDNN$ to this long data. The performance of the $ATNN$ was superior to that of the $TDNN$. The reason for this improvement was that the $ATNN$ has the ability to adapt time-delays in addition to weights, whereas the $TDNN$ only adapts weights (and leaves time-delays fixed). Performance overall was very promising.

The data consisted of three target break-up scenarios, taken from 30°, 60°, and 90° aspect angles. The new data was for the case where the piece can move around on the sensor, so the resulting measurements are noisy. The piece can either be located on one pixel or located over two pixels (half of the piece lies in one pixel and half lies in an adjacent pixel). This is more realistic than assuming the sensor can lock the piece onto one pixel at all times [12]. Thus, the data the neural network is trained and tested on randomly jumps up and down from one time step to the next. This is more realistic than the smooth data we worked with previously. The noise made the recognition problem more challenging for the network.

The $TDNN$ was initially used on this bumpy data and the result was not as good as before (with the smooth data). The $TDNN$ results on the new data are shown in Figures 11, 13, and 15. The $TDNN$ was not capable of resolving two pixel data with slope changes. The $ATNN$ was then trained on the new data and tested. The $ATNN$ provided more flexibility as time-delay elements are adapted in the $ATNN$ according to the characteristics of the data, and better time-delay values are chosen. Several time delays were increased to be over five time steps in the second layer time-delays. The resulting time-delay matrix on the first layer and second layer are shown in Appendix D.

The $ATNN$ achieved 100% correct recognition for all three aspect angles. When the networks were given the entire time (four seconds) to recognize the parts, and the recognition was performed at the end of the data stream, then the $ATNN$ performed with 100% correct recognition whereas the $TDNN$ had only 33% recognition at the end of the data stream.

It is important to consider a scenario where the network is required to identify the parts before the end of the data stream. Thus we computed a stepwise performance percentage, which consists of the percent correct recognition averaged over all possible time steps in the data. On this basis, the $ATNN$ achieved a 99.46% correct recognition rate whereas the $TDNN$ had 85% correct identification over all possible time steps. This means that if the $ATNN$ were requested to make a decision at any time during the incoming data stream, it would have been correct 99% of the time. Early identification is made possible with the $ATNN$. The $ATNN$ performance was far superior to that of the $TDNN$, and was far more resilient to noise in the data. For comparison, the $ATNN$ results (Figures 10, 12, and 14) are shown to the left of the $TDNN$'s results (Figures 11, 13, and 15) on the same observation aspect angles, correspondingly.

3.2 Environmental Variations and Robustness

One of the obscurities that degrades the performance of automatic target recognition is environmental changes and noise. Sources of changes and noise include blurred images, camera vibrations, heat, atmospheric effects, and more. The effects of noise and how to perform recognition in spite of noise will be very important for eventual deployment of this technology. Current target recognition systems are unable to modify their behavior based on

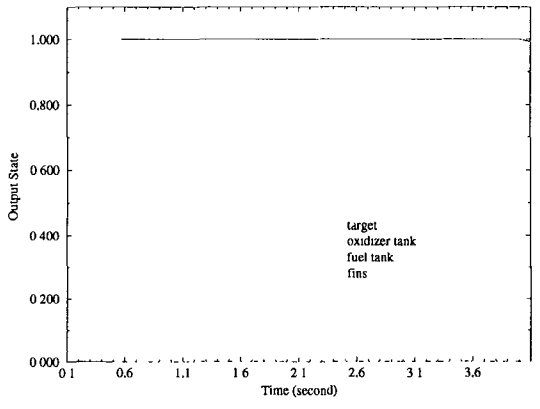


Figure 10: *ATNN* performance on bumpy data with aspect angle 90°, sampling time interval 0.1 sec.

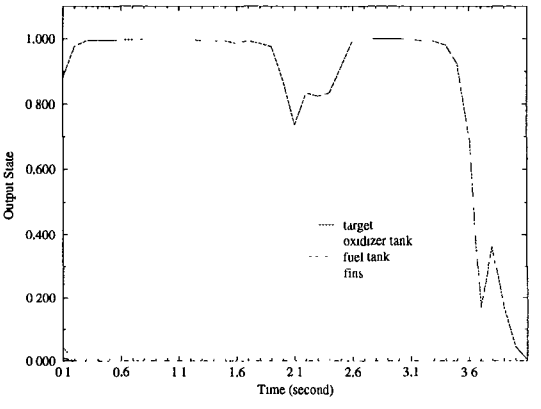


Figure 11: *TDNN* performance on bumpy data with aspect angle 90°, sampling time interval 0.1 sec.

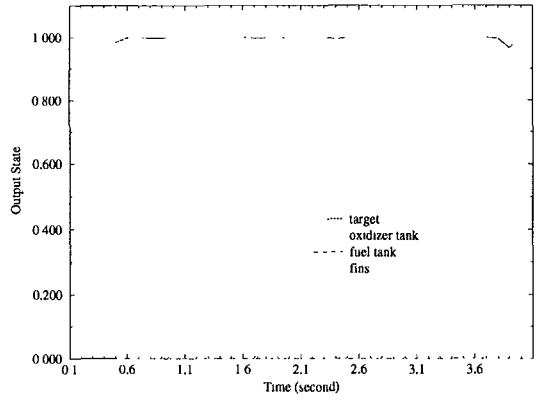


Figure 12: *ATNN* performance on bumpy data with aspect angle 60°, sampling time interval 0.1 sec.

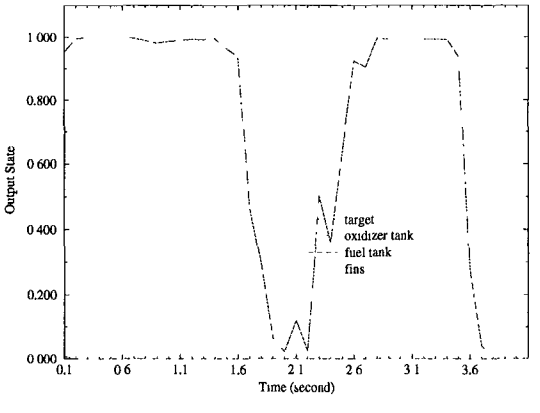


Figure 13: *TDNN* performance on bumpy data with aspect angle 60°, sampling time interval 0.1 sec.

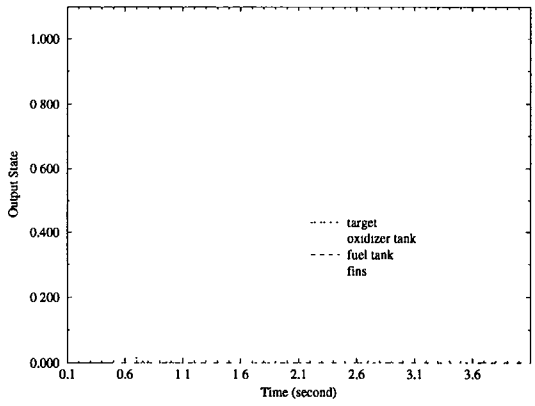


Figure 14: *ATNN* performance on bumpy data with aspect angle 30°, sampling time interval 0.1 sec.

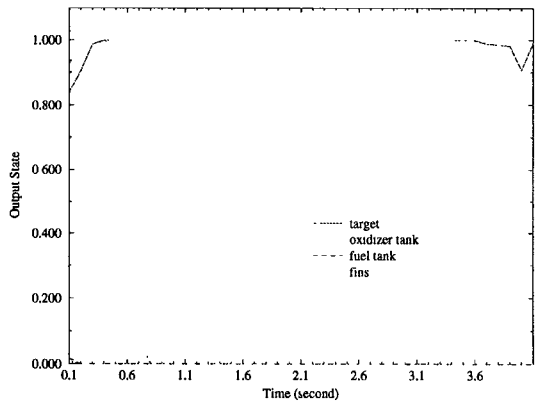


Figure 15: *TDNN* performance on bumpy data with aspect angle 30°, sampling time interval 0.1 sec.

the dynamic environmental changes occurring around them. In order to perform robustness, the system must be able to adapt to this dynamic environment while maintaining acceptable performance [13]. We evaluated our neural network's performance in the presence of noise, and improved the neural network robustness by training on noisy data.

To study the effects of noise on performance, we used varying scenarios in which noise was added during training or recall or both. Simulation results show that the *ATNN* is robust to moderate amounts of noise, and that its robustness improves when training is done with noisy data. This approach is important for eventual implementation in a real world environment, where substantial noise is expected. The data set is based on a simulation of a more realistic scenario than previous data sets, in which the scenario includes tenth second samples with some noise added during simulation by the missile simulator. These initial results showed that the *ATNN* recognized the target parts, in spite of noise during the simulation. We are concerned, however, that the real-world environment would have additional noise; the initial runs did not consider the effects of such additional noise. We address these effects here.

Our evaluation of noisy scenarios first required simulated data consisting of a simulated target scenario. We chose to start with noisy simulated data and to add varying amounts of additional noise. Thus, sources of noise include (1) the simulator or (2) addition of additional noise to simulated data.

Two different types of training scenario were tested:

1. the $ATNN$ was trained with data direct from the missile simulation, and

2. the $ATNN$ was trained with noisy data consisting of simulation data with additional noise added.

The amount of additive noise was varied in each scenario.

The relation of the original raw data (simulated) and the noisy data is shown below:

$$s_n : size + K \times 0.01 \times N_r,$$

$$r_n : radiance + K \times 0.001 \times N_r,$$

where K varies from 1 to 10, and N_r denotes a normal distribution random number with zero mean ($\mu = 0$), and variance (σ) is equal to 1, s_n is an instance of size data with additional noise added, and r_n is an instance of radiance data with additional noise added.

First, the network was trained on simulated data and tested on data with additional noise added. Table 5 shows the results. For each value of K (e.g. each level of additive noise), ten runs were performed. For each value of K used, the ten runs are shown in a column in Table 5. In general, the performance degraded as the amount (K) of noise increased. The data is plotted in Figure 16 with the solid line trace.

runs	K										
	0	1	2	3	4	5	6	7	8	9	10
#1	99.77	100.00	98.64	87.16	75.00	25.00	31.30	25.00	75.00	25.00	25.22
#2	99.77	99.09	95.27	25.00	25.00	75.00	75.00	25.22	75.00	75.22	74.77
#3	99.77	99.77	99.09	91.21	75.00	25.00	75.00	25.00	74.77	25.00	74.32
#4	99.77	99.32	97.52	96.39	76.12	75.00	75.00	25.00	74.77	75.00	36.93
#5	99.77	100.00	99.54	89.86	75.00	75.00	75.00	25.45	80.63	75.00	25.00
#6	99.77	99.32	96.62	85.36	25.00	75.00	84.00	75.00	25.22	75.00	25.00
#7	99.77	99.54	96.62	80.85	83.78	75.00	75.00	75.00	25.22	25.00	75.00
#8	99.77	99.77	98.42	97.74	75.00	68.69	25.00	25.00	75.00	25.00	25.00
#9	99.77	100.00	98.87	92.34	25.00	75.00	25.00	75.00	75.00	26.57	75.00
#10	99.77	99.54	100.00	75.22	75.00	25.00	75.00	25.00	75.00	75.00	76.57
av.	99.77	99.63	98.06	82.11	60.99	59.36	61.53	40.06	65.56	50.18	51.28

Table 5: Performance of $ATNN$ trained on pure data and tested with varying amounts of noise added.

Figure 16 shows the property of graceful degradation, as performance is quite high (98-100%) until $K > 2$.

Next the network was trained with data that included additive noise. This data set contained the original simulated data in addition to the data with additive noise. The variance values of the additive noise were 0.02 and 0.002 for size and radiance respectively.

The trained $ATNN$ was again tested on varying amounts of noisy data as described above. The results of ten runs were shown in Table 6. The average performance is provided in the last row. Compared with the average performance in Table 5 (the last row), the network trained with additive noise improved the identification capability. The comparison is shown in Figure 16.

For low noise ($K \leq 2$), there was slightly lower performance when the network was trained on noisy data. For higher noise ($2 < K < 8$), better performance was attained when the network was trained with additive noise. For even larger amounts of noise ($K \geq 8$), there

runs	K										
	0	1	2	3	4	5	6	7	8	9	10
#1	94.94	93.18	96.71	96.21	87.62	72.47	77.02	73.48	30.30	69.44	66.66
#2	94.94	94.69	92.17	79.79	72.22	54.79	30.05	31.31	50.25	70.95	26.76
#3	94.94	95.95	95.95	96.21	64.39	72.72	29.79	73.48	72.22	77.02	59.34
#4	94.94	95.45	95.45	94.19	87.12	66.66	78.28	28.28	71.46	30.80	66.16
#5	94.94	96.96	96.96	91.41	80.30	83.83	77.77	75.25	73.48	39.14	61.61
#6	94.94	94.19	94.69	90.65	86.86	32.82	80.80	71.46	65.90	31.56	68.93
#7	94.94	94.44	93.18	92.17	90.40	41.91	57.82	76.01	72.72	69.19	28.53
#8	94.94	94.44	95.45	95.20	28.53	84.09	73.98	71.21	27.27	67.42	70.20
#9	94.94	95.45	95.20	94.69	83.58	63.38	71.21	26.26	30.30	65.65	35.85
#10	94.94	94.44	96.46	85.60	83.33	71.96	30.30	66.41	31.56	53.78	76.01
av.	94.94	94.92	95.22	91.61	76.43	64.46	60.70	59.31	52.55	57.50	56.01

Table 6: Performance of $ATNN$ trained on pure and noisy data and tested on varying amounts of noise added.

was less jitter in performance when the network was trained with additive noise – more stable performance. We can conclude that the $ATNN$ is more robust in noise situations when it is trained with additive noise. Similar studies and conclusions can be found in [5]. The lower range of noise, where the non-additive networks performed better, are not likely to occur in the real world.

4 Discussion

High performance in target component discrimination is presented here using a neural network approach. The $TDNN$ and $ATNN$ were shown promising in recognizing targets regardless of different target types, different aspect angles, tumbling situations or broken pieces. The recognition performance of the $ATNN$ outstrips that of the $TDNN$. The $ATNN$ further possesses much earlier discrimination and noise resilience. We have sug-

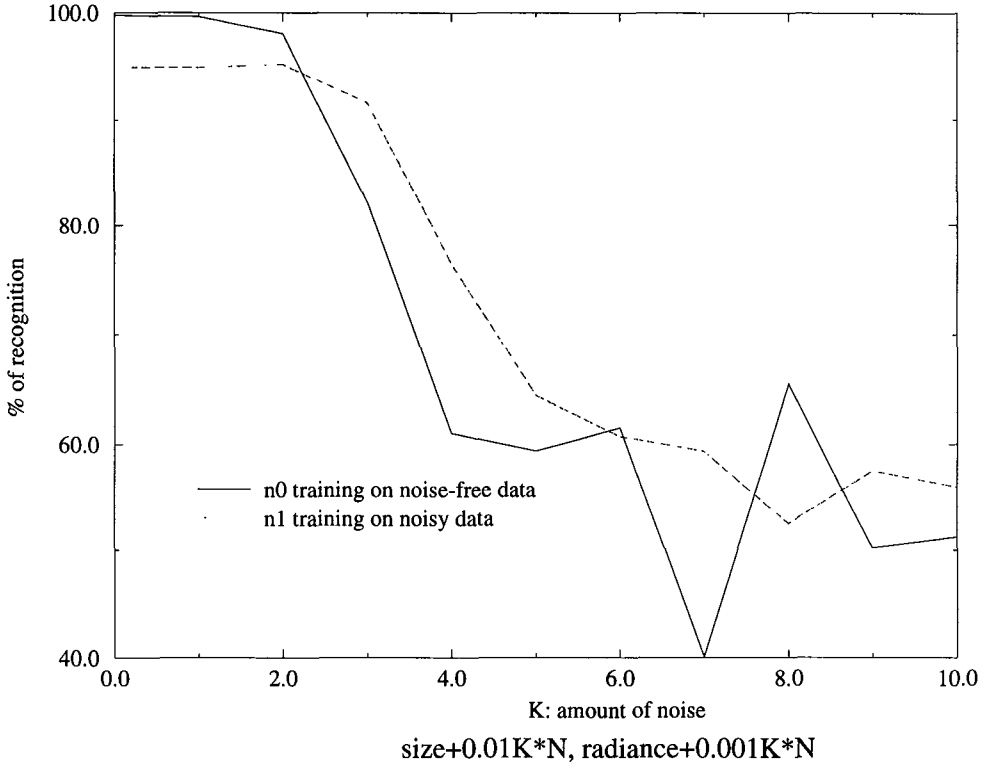


Figure 16: Performance of the $ATNN$ as a function of the amount of noise in the data.For noisy environments, performance is superior when training is done on noisy data.

gested that the $ATNN$ is well-suited for spatiotemporal and temporal domains of problems due to its capability of handling multiple channel data and correlating relationships among these data [6].

Specifically, we have shown that for a threat in an exoatmospheric trajectory, and with a single sensor, the following is the case:

1. We have previous results that the TDNN is capable of high performance for discrimination of warhead versus other missile parts.

2. Performance can be improved by appropriate scaling and training schedule.

3. Higher sampling rate data gave better discrimination performance.

4. Training the TDNN with adaptive time delays (i.e. as an ATNN) enhanced performance in the trained networks.

5. Training with noise produced a network that is more robust to variations in real-world scenarios.

Future work should include extending the discrimination approach by developing one or more neural networks that can identify a warhead regardless of the type of threat.

Appendix

A Simulation Results on 0.5 Second Data

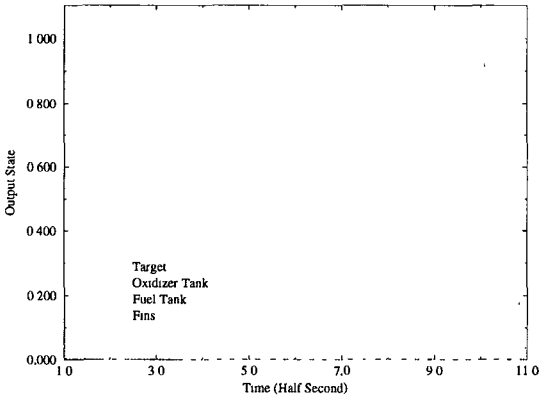


Figure 1: Recognizing target in trajectory 1 with aspect angle 30°.

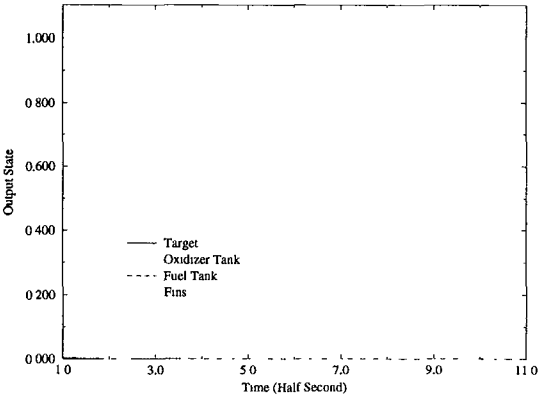


Figure 2: Recognizing target in trajectory 1 with aspect angle 45°.

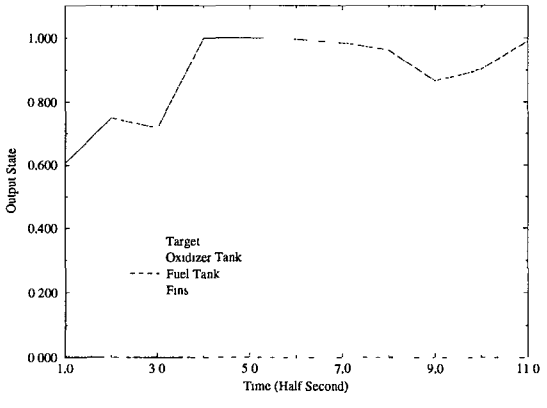


Figure 3: Recognizing target in trajectory 1 with aspect angle 60°.

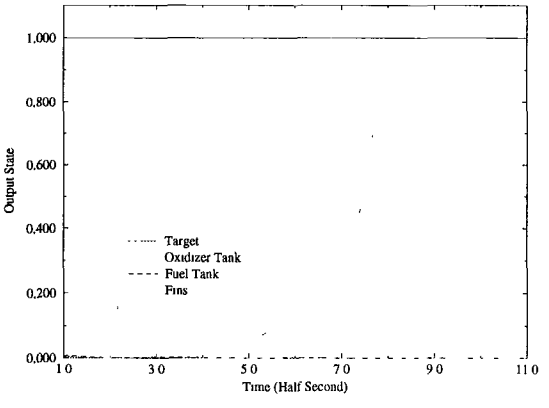


Figure 4: Recognizing target in trajectory 1 with aspect angle 75°.

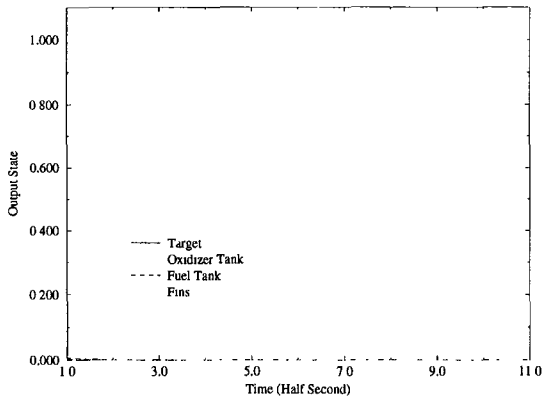


Figure 5: Recognizing target in trajectory 1 with aspect angle 90°.

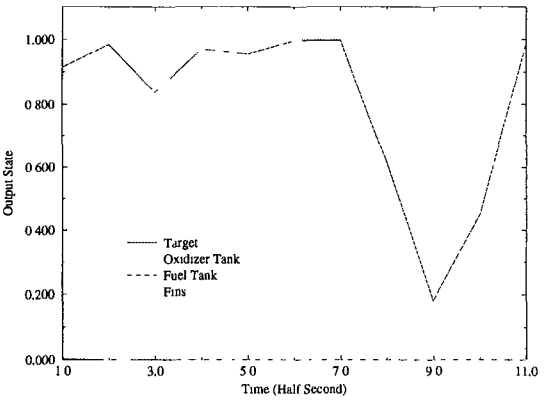


Figure 6: Recognizing target in trajectory 2 with aspect angle 30°.

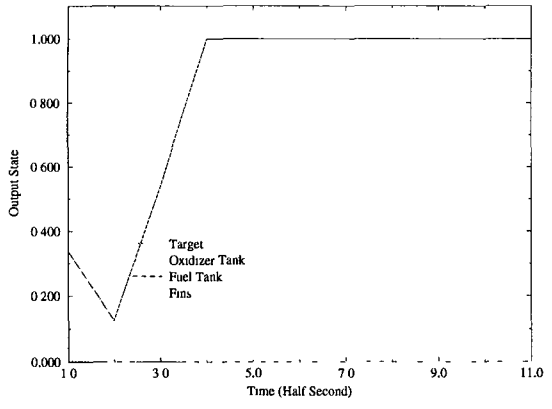


Figure 7: Recognizing target in trajectory 2 with aspect angle 60°.

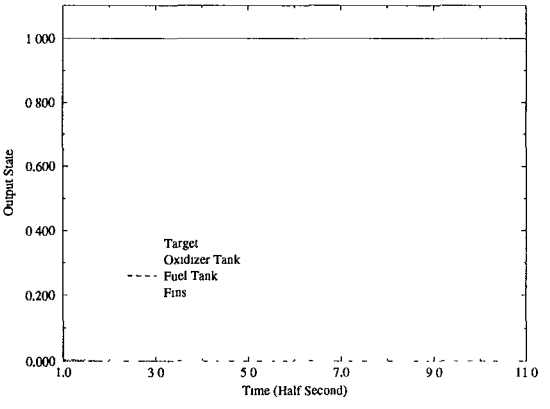


Figure 8: Recognizing target in trajectory 2 with aspect angle 90°.

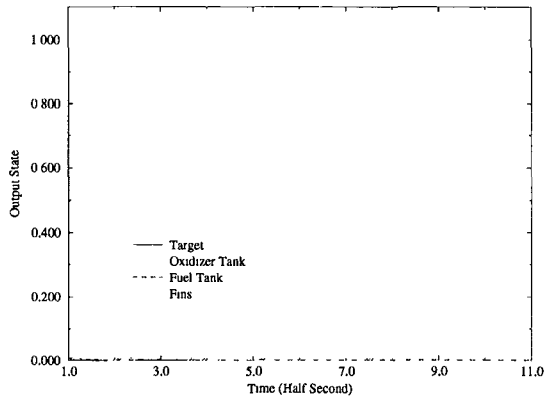


Figure 9: Recognizing target with aspect angle 90°, front $\frac{1}{3}$ component 2 replaces whole component 2.

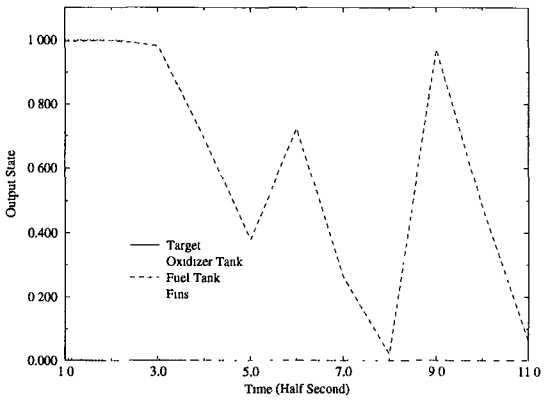


Figure 10: Recognizing target with aspect angle 90° with back $\frac{1}{3}$ component 2 replaces whole component 2.

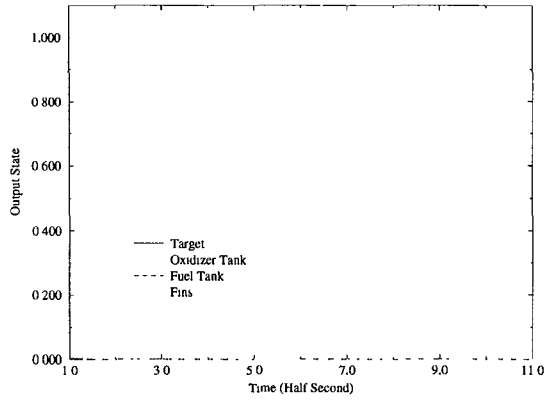


Figure 11: Recognizing target with aspect angle 90° with front $\frac{1}{2}$ component 3 replaces whole component 3.

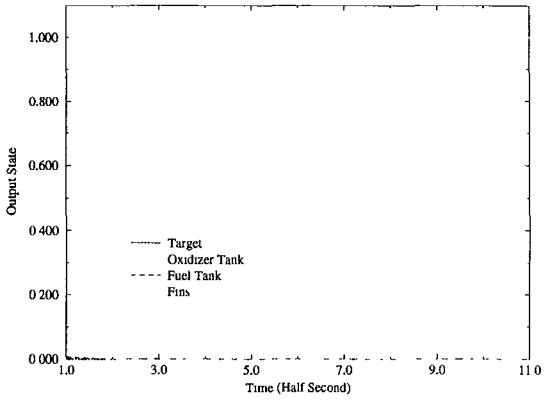


Figure 12: Recognizing target with aspect angle 90° with front $\frac{1}{3}$ component 4 replaces whole component 4.

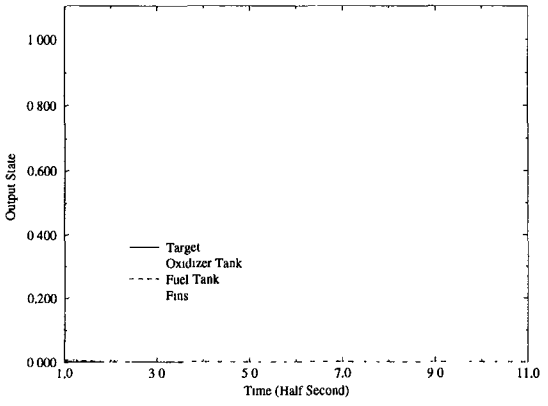


Figure 13: Recognizing target with components tumbling angles: 90°, 60°, 90°, and 90° respectively.

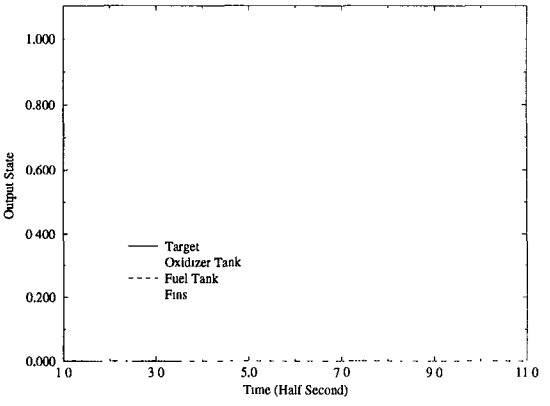


Figure 14: Recognizing target with components tumbling angles: 90°, 60°, 60°, and 90° respectively.

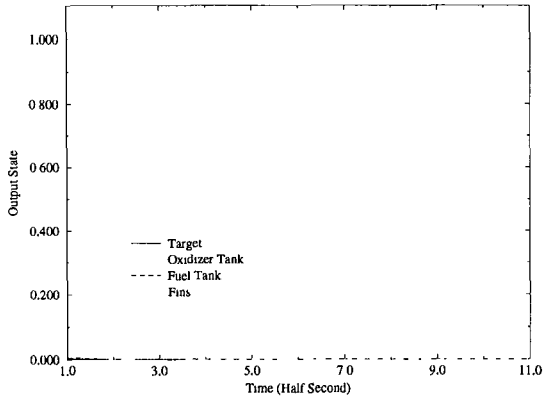


Figure 15: Recognizing target with components tumbling angles: 90°, 60°, 60°, and 30° respectively.

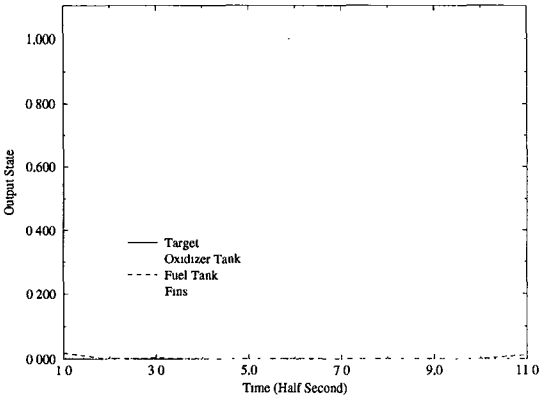


Figure 16: Recognizing target with components tumbling angles: 60°, 90°, 60°, and 60° respectively.

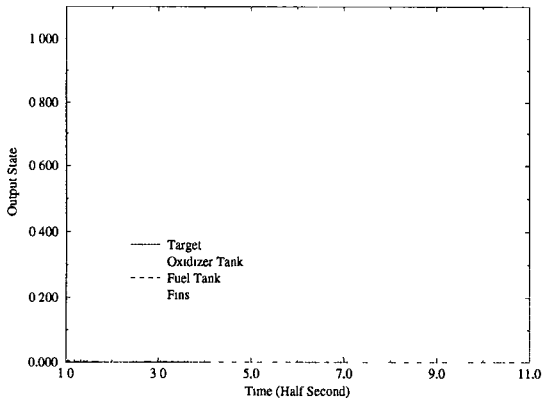


Figure 17: Recognizing target with components tumbling angles: 60°, 90°, 30°, and 60° respectively.

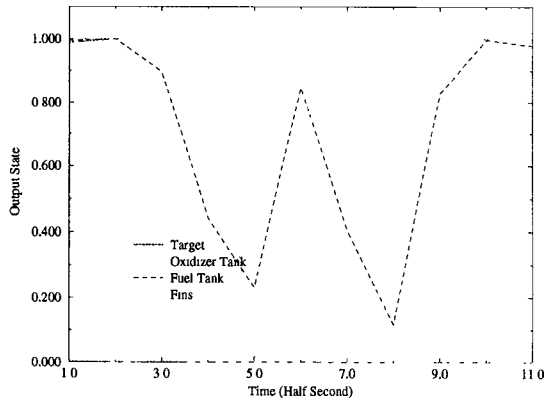


Figure 18: Recognizing target with components tumbling angles: 60°, 90°, 90°, and 60° respectively.

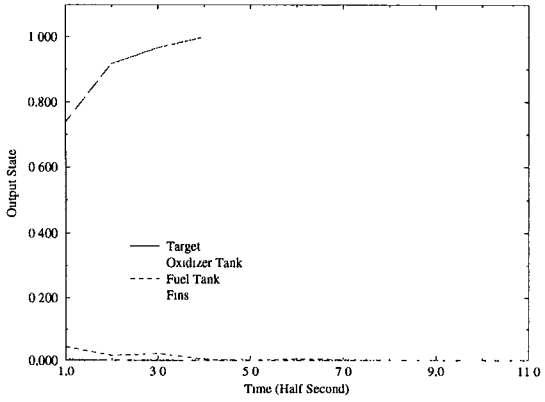


Figure 19: Recognizing target with components tumbling angles: 60°, 30°, 30°, and 60° respectively.

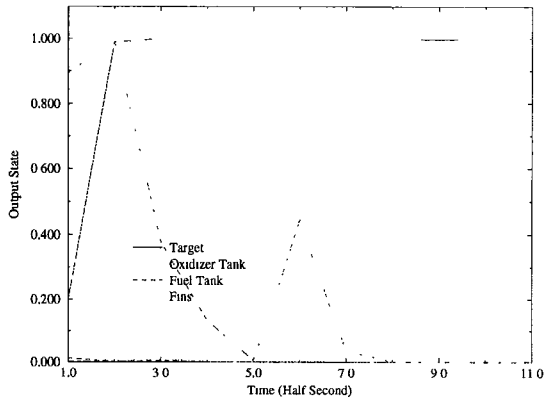


Figure 20: Recognizing target with components tumbling angles: 60°, 30°, 60°, and 90° respectively.

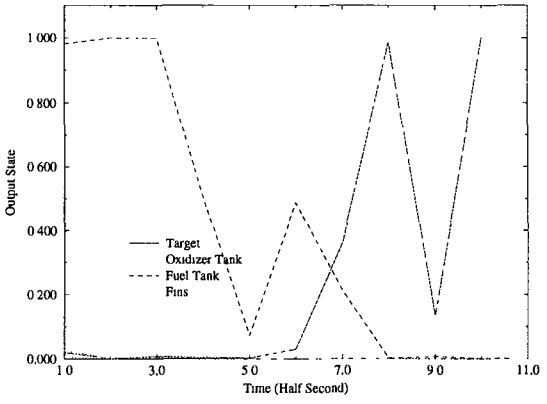


Figure 21: Recognizing target with components tumbling angles: 60°, 30°, 90°, and 90° respectively.

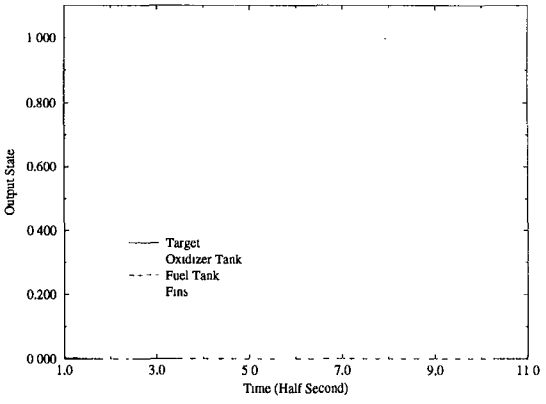


Figure 22: Recognizing target with components tumbling angles: 30°, 60°, 30°, and 30° respectively.

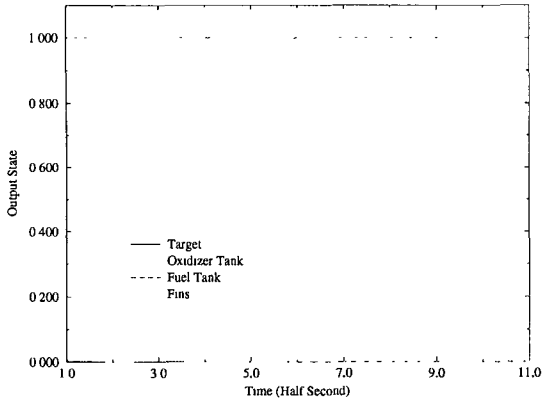


Figure 23: Recognizing target with components tumbling angles: 30°, 60°, 90°, and 30° respectively.

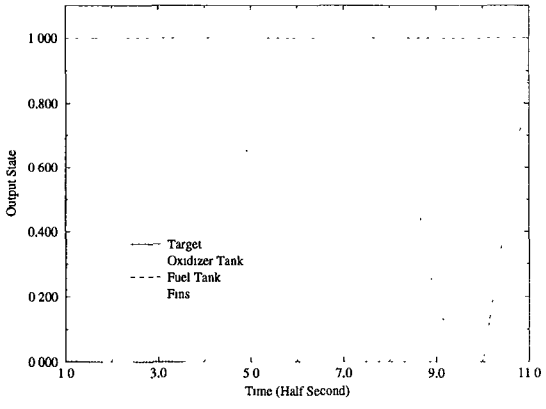


Figure 24: Recognizing target with components tumbling angles: 30°, 60°, 90°, and 60° respectively.

B More Results on 0.1 Second Data

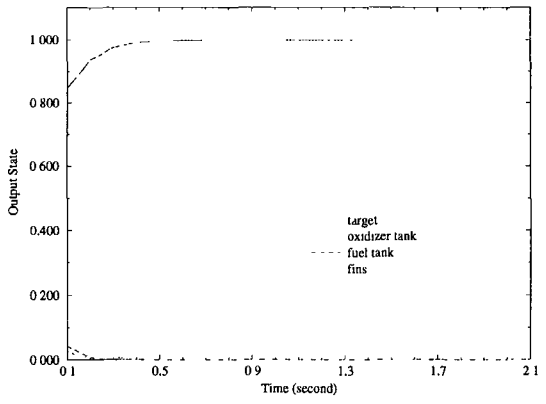


Figure 1: Recognizing target with aspect angle 90°.

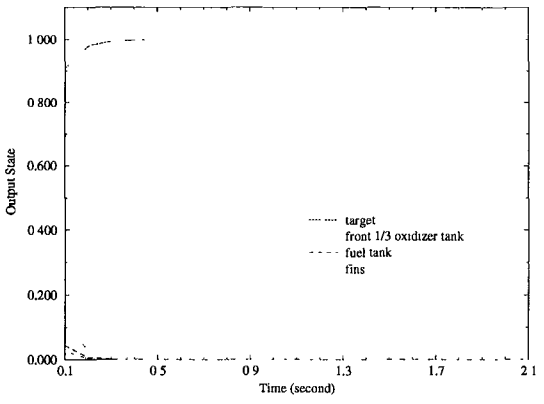


Figure 2: Recognizing target with aspect angle 90°, front 1/3 component 2 replaces the whole.

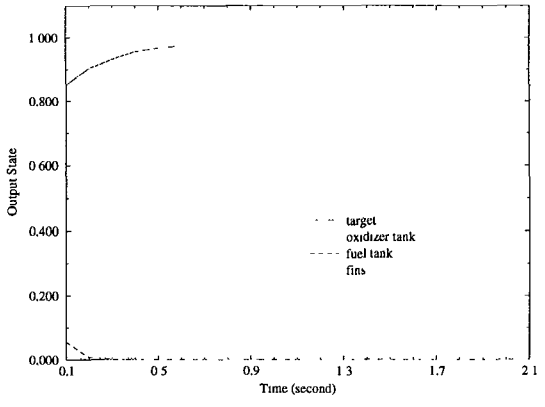


Figure 3: Recognizing target with aspect angle 90°, components tumbling angle: 60°, 90°, 90°, and 30° respectively.

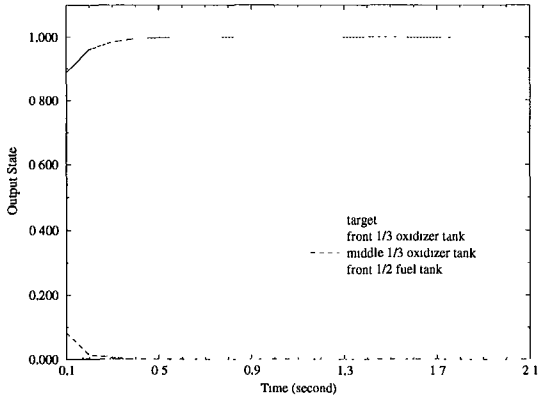


Figure 4: Recognizing target with aspect angle 90°, front 1/3 component 2 middle 1/3 component 3, front 1/2 component 4.

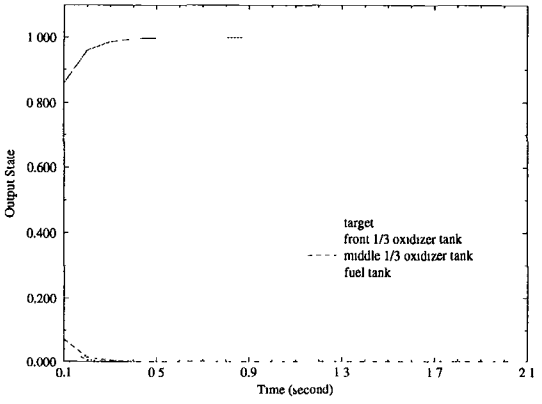


Figure 5: Recognizing target with aspect angle 90°, front 1/3 component 2, middle 1/3 component 2, and component 4

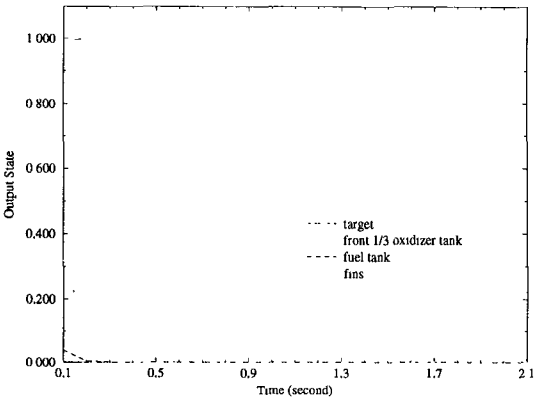


Figure 6: Recognizing target with aspect angle 30°, front 1/3 component 2 replaces whole component 2.

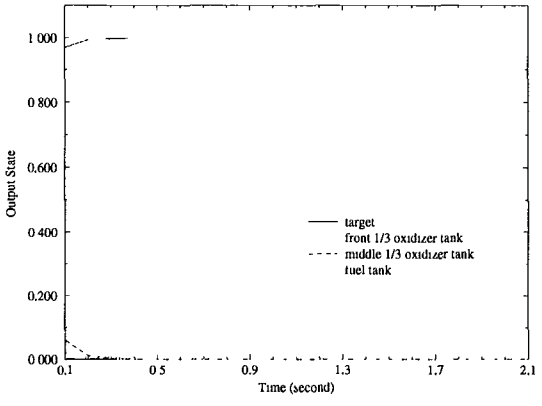


Figure 7: Recognizing target with aspect angle 30°, front 1/3 component 2, middle 1/3 component 2, component 3

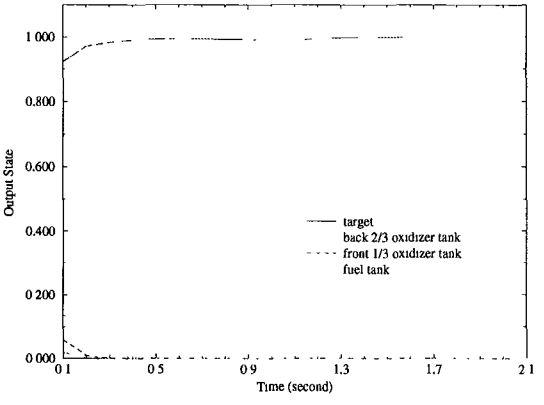


Figure 8: Recognizing target with aspect angle 30°, front 1/3 component 2, back 2/3 component 2, and component 4.

C Simulation Results on partial 0.1 second data.

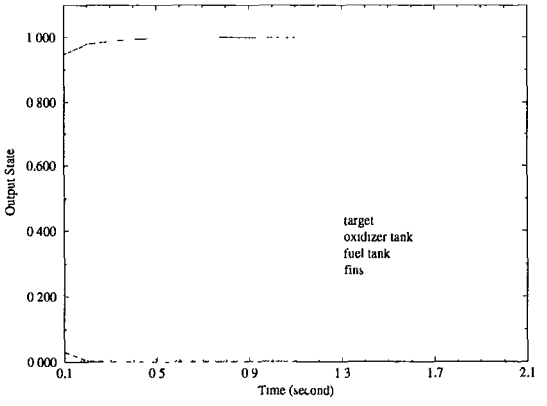


Figure 1: Aspect angle 30°.

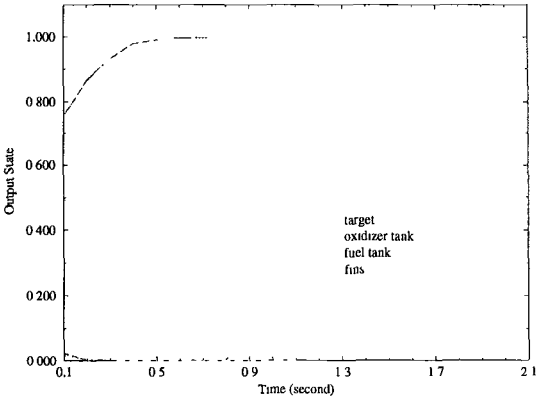


Figure 2: Aspect angle 60°.

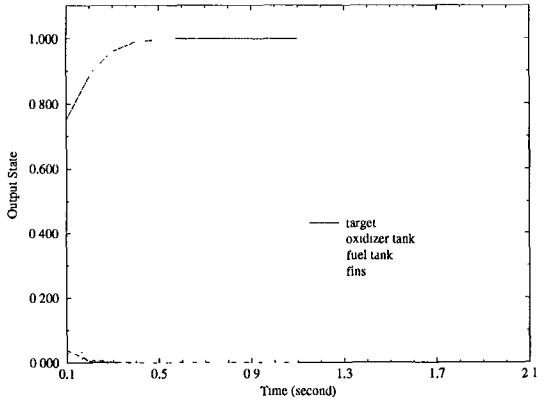


Figure 3: Aspect angle 90°.

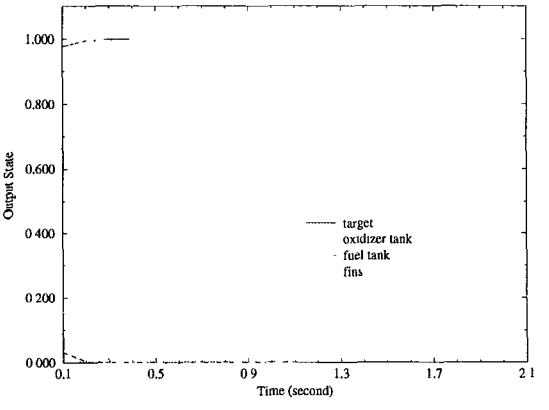


Figure 4: Aspect angle 30°, front 1/3 component 2 replaces whole component 2.

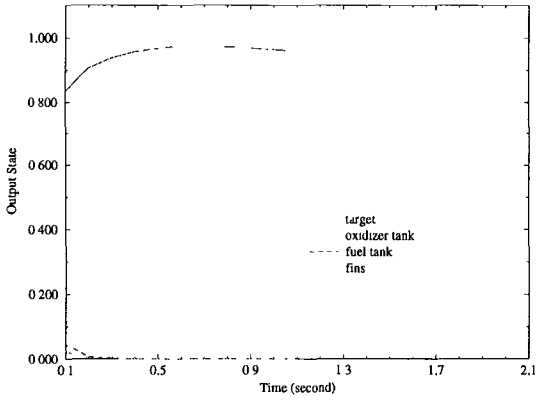


Figure 5: Aspect angle 30°, back 1/3 component 2 replaces whole component 2, front 1/3 component 2 replaces whole component 3.

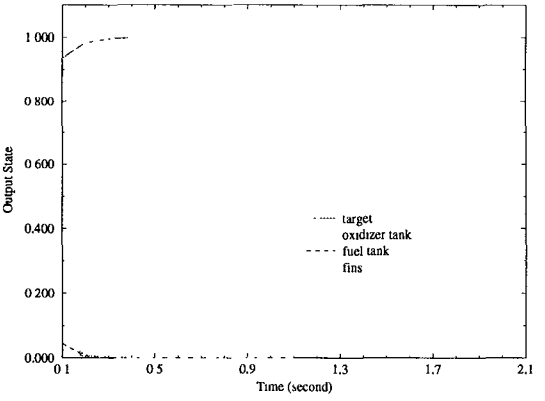


Figure 6: Aspect angle 30°, front 1/3 component 2 replaces whole component 2, middle 1/3 component 2 replaces whole component 3.

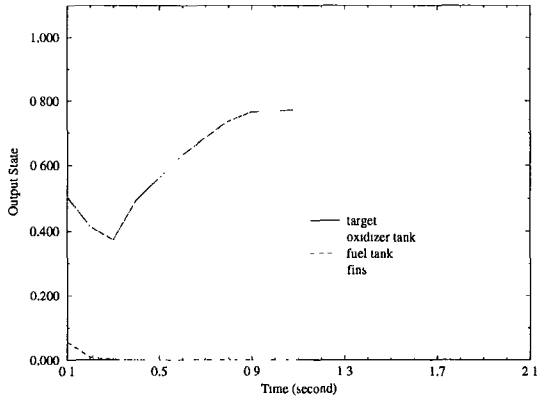


Figure 7: Aspect angle 90°, with components tumbling angle: 30°, 60°, 90°, and 30° respectively.

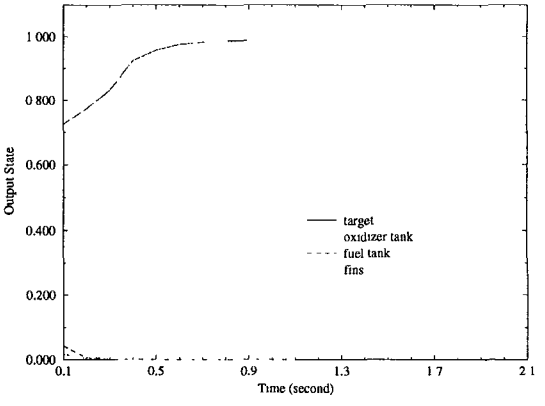


Figure 8: Aspect angle 90°, with components tumbling angle: 60°, 90°, 90°, and 30° respectively.

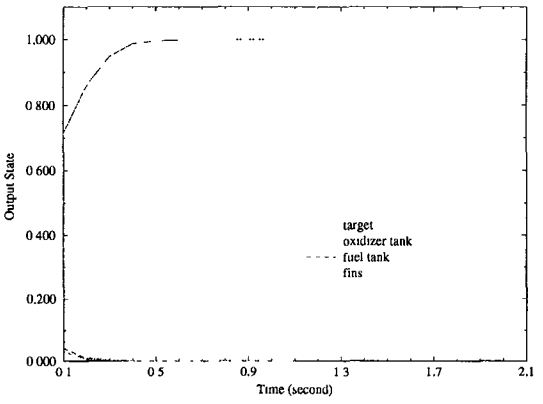


Figure 9: Aspect angle 90°, time 323-325 second.

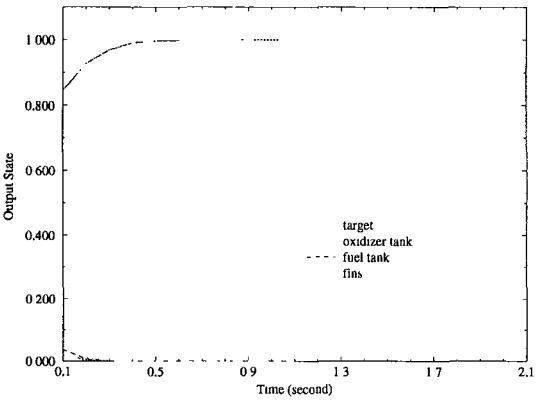


Figure 10: Aspect angle 90°, with components tumbling angle: 90°, 60°, 60°, and 30° respectively.

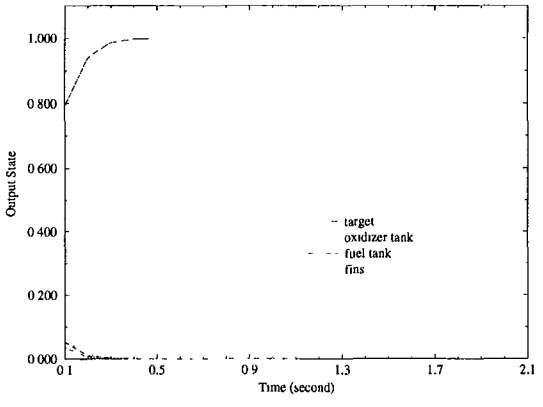


Figure 11: Aspect angle 90°, front 1/3 component 2 replaces whole component 2, middle 1/3 component 2 replaces whole component 3.

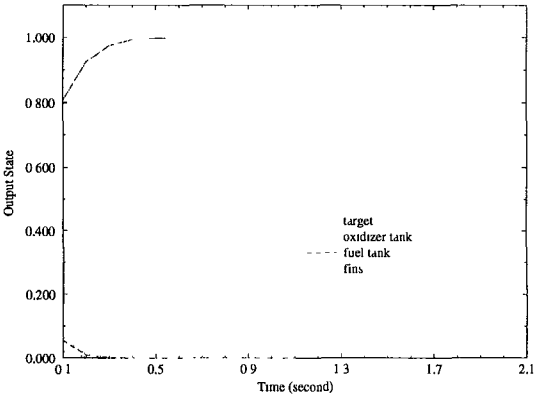


Figure 12: Aspect angle 90°, front 1/3 component 2 replaces whole component 2, middle 1/3 component 2 replaces whole component 3, front 1/2 component 3 replaces component 4.

D Resulting ATNN Time-Delay Matrix

T1 is the time-delay matrix on the first layer and T2 is the time-delay matrix for the second layer.

$$T1 = \begin{bmatrix} 1 & 1 & 2 & 3 & 1 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \\ 1 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 & 0 & 1 & 2 & 3 \end{bmatrix}$$

$$T2 = \begin{bmatrix} 0 & 1 & 2 & 3 & 4 & 9 & 0 & 1 & 2 & 3 & 4 & 5 & 0 & 1 & 2 & 3 & 4 & 5 \\ 0 & 1 & 2 & 3 & 4 & 6 & 0 & 1 & 2 & 3 & 4 & 5 & 0 & 1 & 2 & 3 & 4 & 5 \end{bmatrix}$$

References

[1] B. Bai and N.H. Farhat. Learning networks for extrapolation and radar target identification. *Neural Networks*, 5:507–529, 1992.

[2] S. P. Day and M. Davenport. Continuous-time temporal back-propagation with adaptive time delays. Neuroprose archive, Ohio State University. Accessible on Internet via anonymous ftp on archive.cis.ohio-state.edu, in pub/neuroprose/day.tempora.ps August, 1991.

[3] S. P. Day and M. R. Davenport. Continuous-time temporal back-propagation with adaptive time delays. *IEEE Trans. on Neural Networks*, 4(2):348–354, March 1993.

[4] R.P. Gorman and T.J. Sejnowski. Analysis of hidden units in a layered network trained to classify sonar targets. *Neural Networks*, 1:75–89, 1988.

[5] L. Holmström and P. Koistinen. Using additive noise in back-propagation training. *IEEE Trans. on Neural Networks*, 3(1):24–38, January 1992.

[6] D.-T. Lin. *The Adaptive Time-Delay Neural Network: Characterization and Applications to Pattern Recognition, Prediction, and Signal Processing.* PhD thesis, University of Maryland at College Park, 1994.

[7] D.-T. Lin, J. E. Dayhoff, and P. A. Ligomenides. Adaptive time-delay neural network for temporal correlation and prediction. In *Intelligent Robots and Computer Vision XI: Biological, Neural Net, and 3-D Methods, Proc. SPIE*, volume 1826, pages 170–181, Boston, November, 1992.

[8] D.-T. Lin, J. E. Dayhoff, and P. A. Ligomenides. A learning algorithm for adaptive time-delays in a temporal neural network. Technical Report SRC-TR-92-59, Systems Research Center, University of Maryland, College Park, Md 20742, May 15 1992.

[9] D.-T. Lin, J. E. Dayhoff, and P. A. Ligomenides. Trajectory recognition with a time-delay neural network. In *International Joint Conference on Neural Networks*, volume 3, pages 197–202, Baltimore, 1992. IEEE, New York.

[10] J.L. McClelland, D.E. Rumelhart, and the PDP Research Group. *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, volume 2. MIT Press, Cambridge, 1986.

[11] C.L. Resch. New time delay neural network to distinguish exo-atmospheric warheads. Technical Report AM-93-E141, Applied Physics Laboratory, The Johns Hopkins University, Laurel, MD, August 1993.

[12] C.L. Resch. Effects of jitter on the ability of a time delay neural network to distinguish exo-atmospheric warheads. Technical Report AM-94-E010, Applied Physics Laboratory, The Johns Hopkins University, Laurel, MD, January 1994.

[13] M.W. Roth. Survey of neural network technology for automatic target recognition. *IEEE Trans. on Neural Networks*, 1(1):28–43, March 1990.

[14] K.P. Unnikrishnan, J.J. Hopfield, and D.W. Tank. Connected-digit speaker-dependent speech recognition using a neural network with time-delayed connections. *IEEE Trans. on Signal Processing*, 39(3):698–713, 1991.

[15] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition: Neural networks versus hidden markov models. In *Proc. IEEE Int. Conf. Acoust., Speech, Signal Processing*, pages 107–110, April 1988.

[16] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang. Phoneme recognition using time-delay neural networks. *IEEE Trans. on Acoust., Speech, Signal Processing*, 37(3):328–339, 1989.

[17] A. Waibel, K. J. Lang, and G. E. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3:23–43, 1990.