The Truck Backer-Upper

Nguyen & Widrow (1990)

Setup

mitted. The specific problem treated in this paper is that of the design by self-learning of a nonlinear controller to control the steering of a trailer truck while backing up to a loading dock from an arbitrary initial position. Only backing up is allowed. Computer

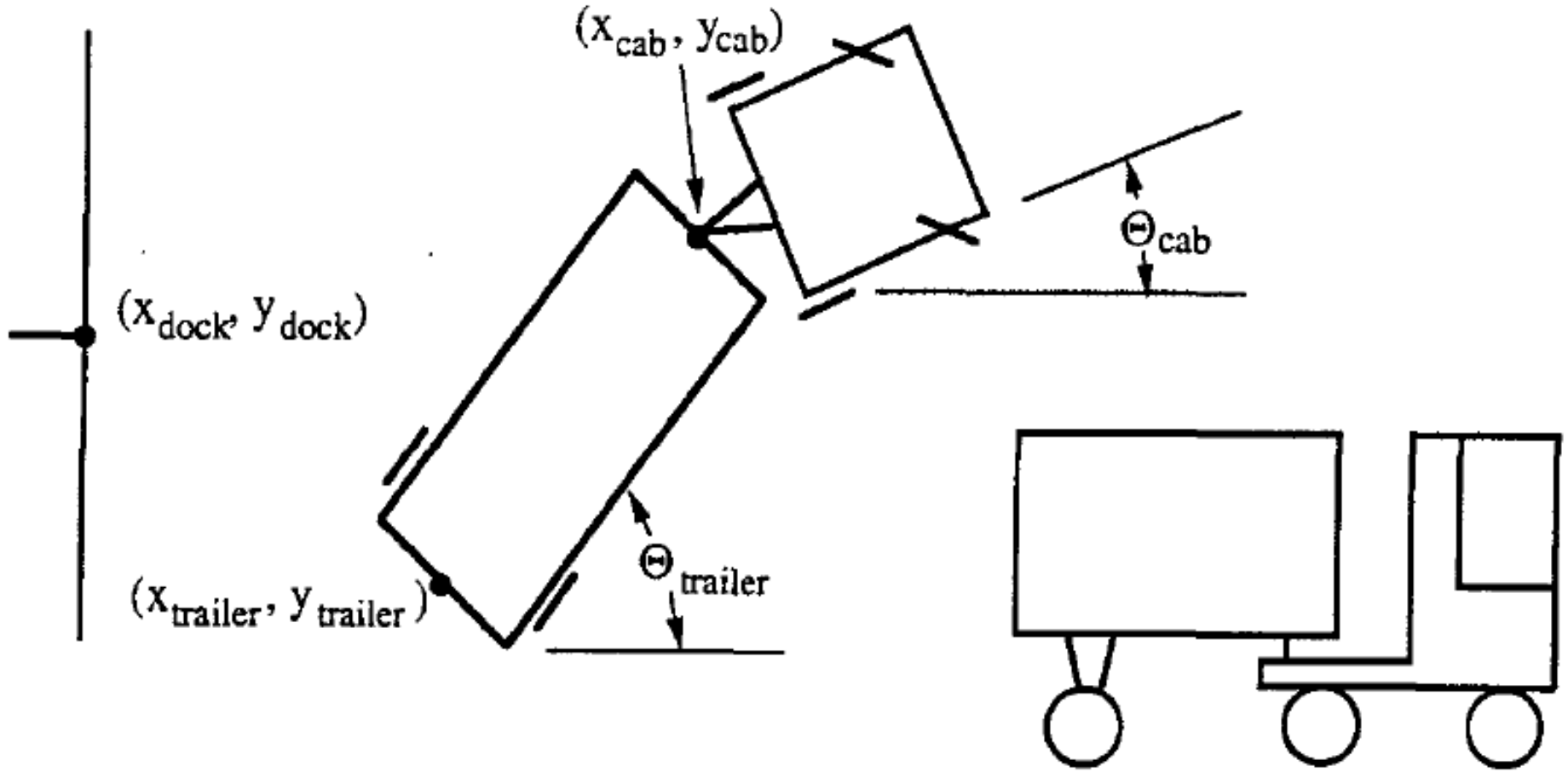


Figure 1: The truck, the trailer, and the loading dock

state variables representing the position of the truck and that of the loading dock are θ_{cab}, the angle of the truck, x_{cab} and y_{cab}, the cartesian position of the yoke, $x_{trailer}$ and $y_{trailer}$, the cartesian position of the rear of the center of the trailer, θ_{trailer}, the angle of the trailer

The truck backs up until it hits the dock, then stops. The goal is to cause the back of the trailer to be parallel to the loading dock, and to have the point $(x_{trailer}, y_{trailer})$ be aligned as closely as possible with point (x_{dock}, y_{dock}). The controller will learn

Training

volves a two-stage learning process. The first stage involves the training of a neural network to be an emulator of the truck and trailer kinematics. The second stage involves the training of a neural-network controller to control the emulator. A similar ap-

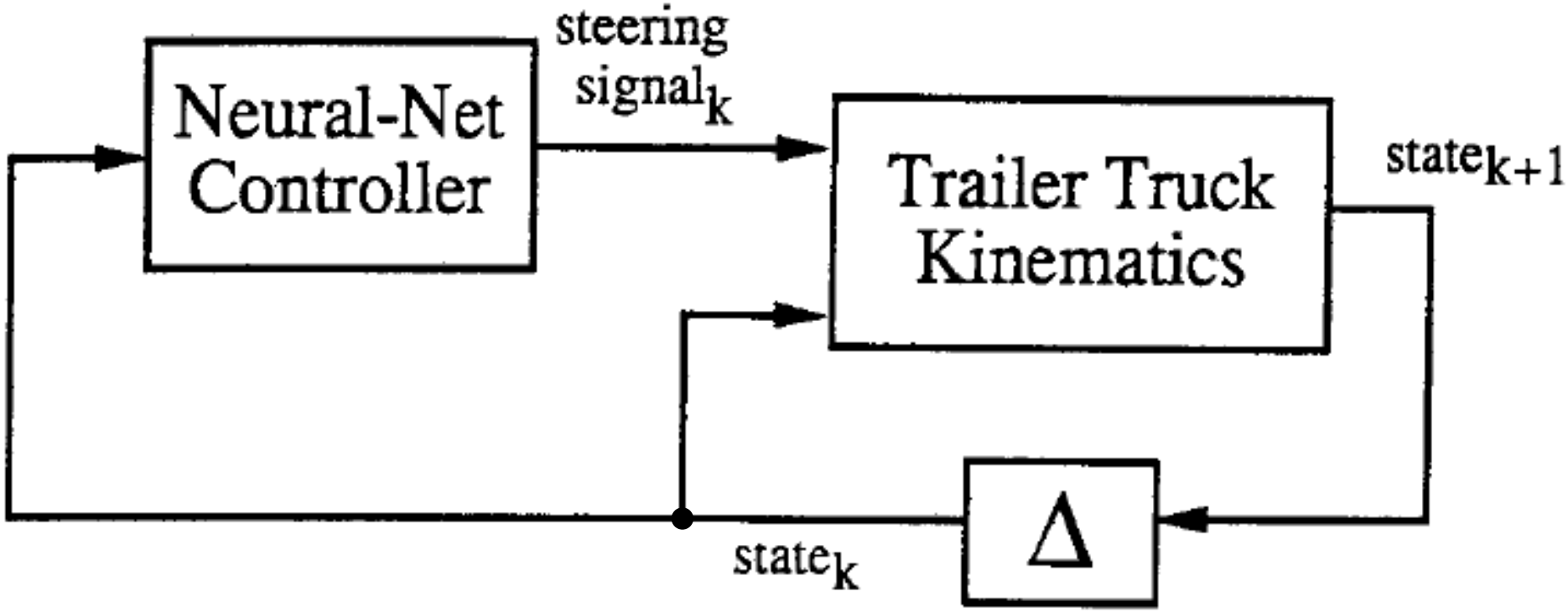
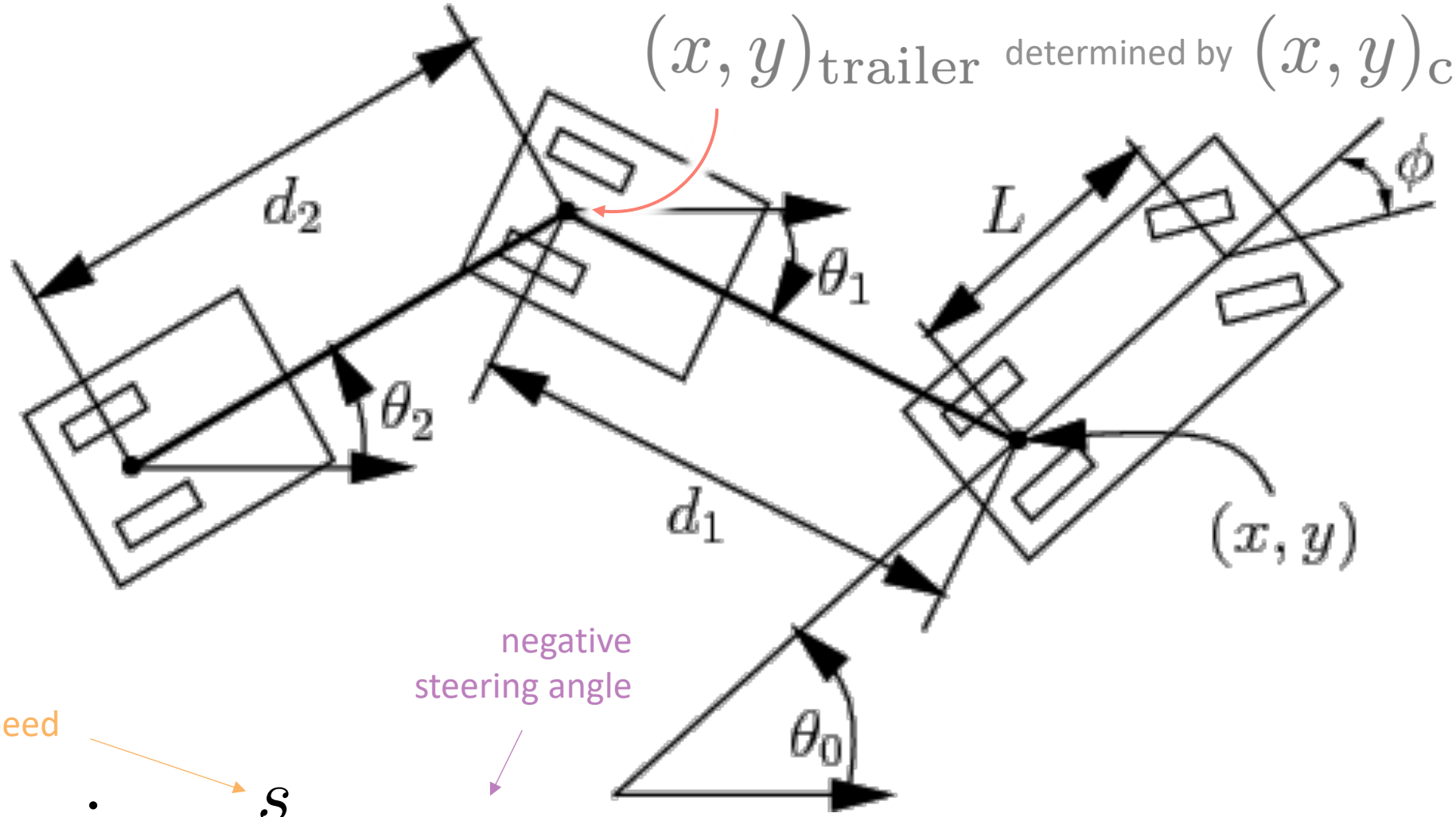


Figure 2: Overview Diagram

present state vector $state_k$ is fed to the controller which in turn provides a $steering\ signal_k$ between -1 (hard right) and $+1$ (hard left) to the truck. The time index is k. Each time cycle, the truck backs up by a fixed small distance. The next state is deter-

$(x, y)_{\text{trailer}}$ determined by $(x, y)_{\text{cab}}, d_1, \theta_1$



negative steering angle

signed speed

$$\dot{\theta}_0 = \frac{s}{L} \tan \phi$$

$$\dot{\theta}_1 = \frac{s}{d_1} \sin(\theta_1 - \theta_0)$$

$$\dot{x} = s \cos \theta_0$$

$$\dot{y} = s \sin \theta_0$$

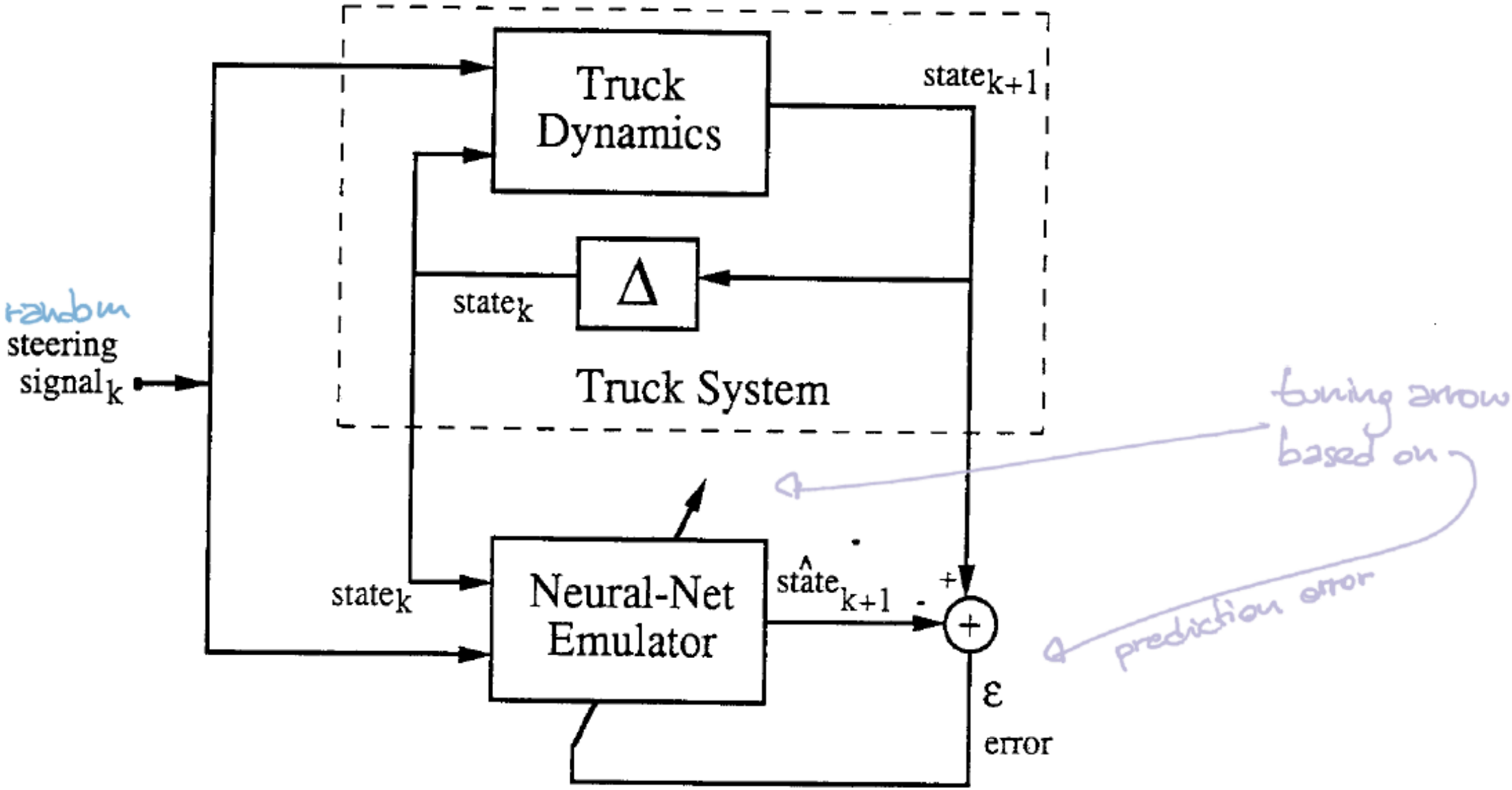


Figure 3: Training the neural-net truck emulator

Figure 3 shows a block diagram of the process used to train the emulator. The truck backs up randomly, going through many cycles with randomly se-

The emulator, chosen as a two layer neural network, learns to generate the next positional state vector when given the present state vector and the steering signal. This is done for a wide variety of positional

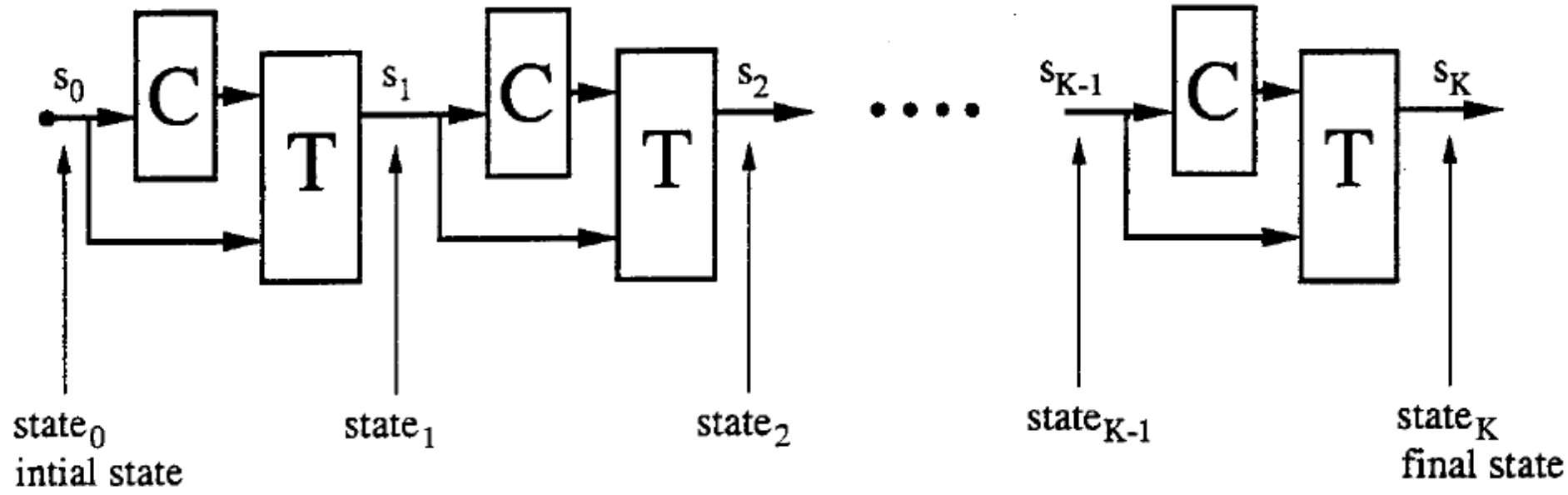
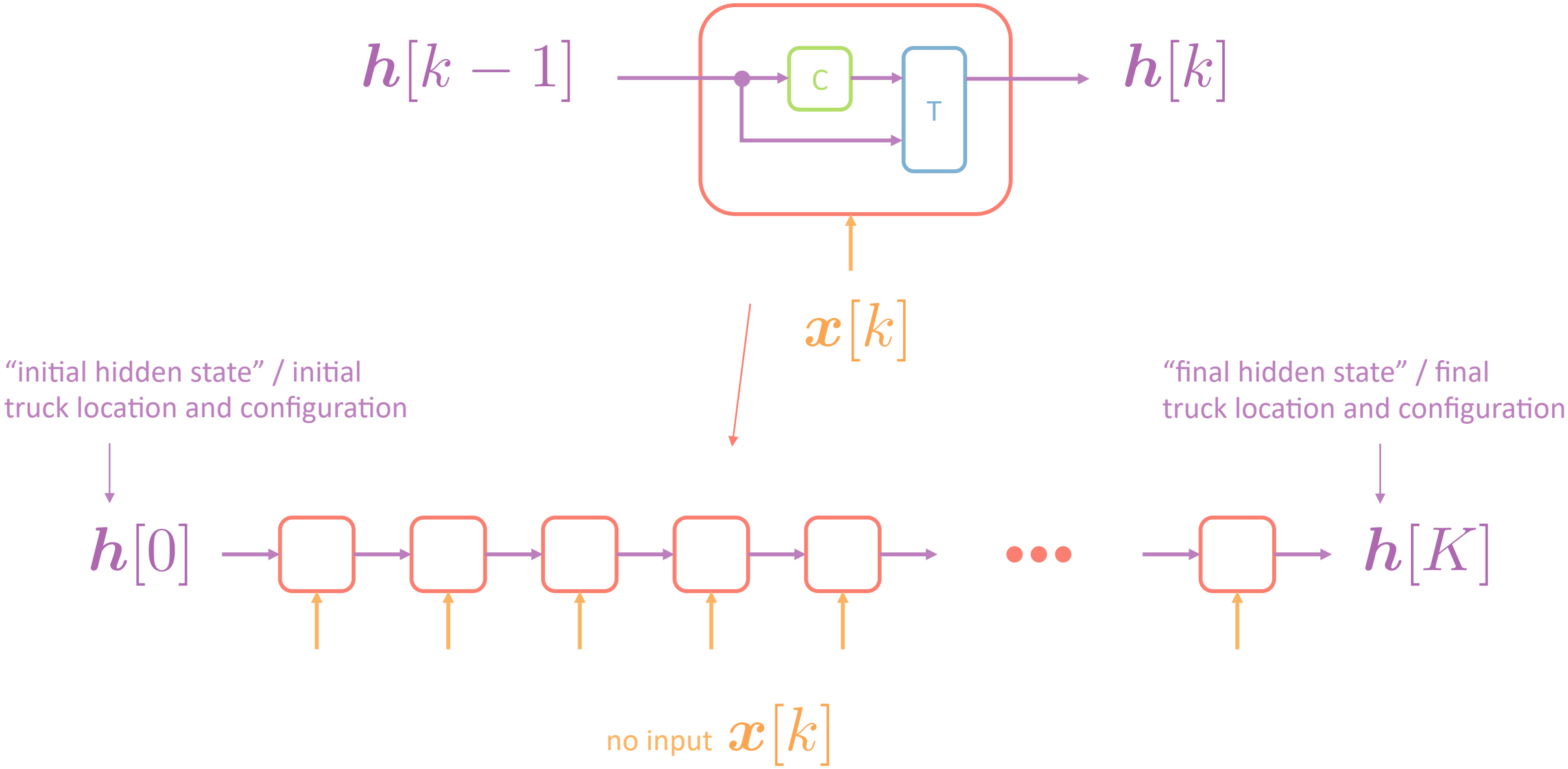
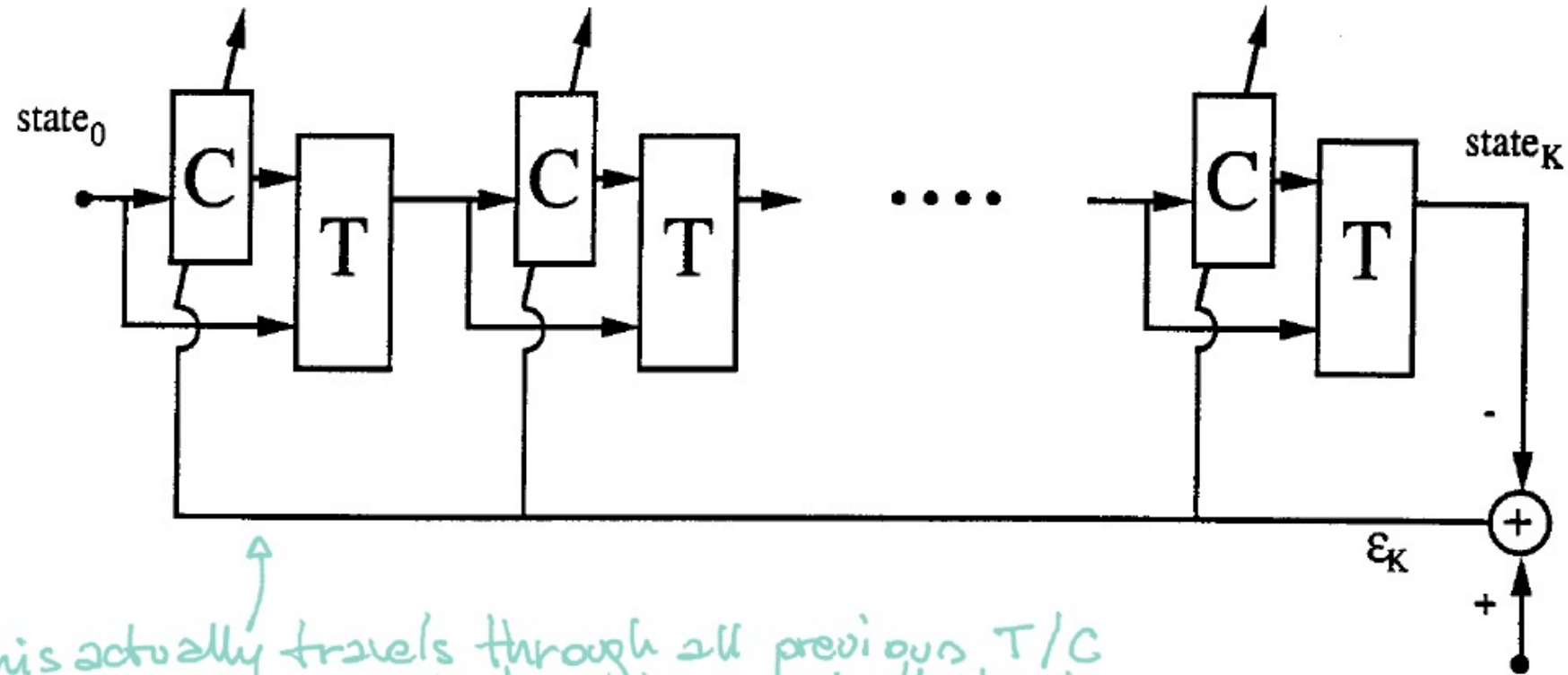


Figure 4: State transition flow diagram

Refer to Figure 4. The identical blocks labeled C represent the controller. The identical blocks labeled T represent the truck and trailer emulator. Suppose


$\boldsymbol{h}[k-1]$
C
T
$\boldsymbol{h}[k]$
$\boldsymbol{x}[k]$
"initial hidden state" / initial truck location and configuration
"final hidden state" / final truck location and configuration
$\boldsymbol{h}[0]$
$\boldsymbol{h}[K]$
no input $\boldsymbol{x}[k]$



This actually travels through all previous T/C modules. The visualisation shows only that all C blocks are updated (also) proportionally to the final error (which ~~assumes~~ implies a MSE loss).

Final desired response:

$x_{trailer} = x_{dock}$

$y_{trailer} = y_{dock}$

$\Theta_{trailer} = 0$

Figure 5: Training the controller with back-propagation

The initial position of the truck is chosen at random. The truck backs up, undergoing many individual back up cycles, until it stops. The final error is used by back-propagation to adapt the con-

Back-prop through time with variable unrolling period K

dent of each other, but the actual weight changes in C are taken as the sum of the tentative changes.

entire process is repeated by placing the truck and trailer in another initial position, and allowing it to back up until it stops. Once again, the controller

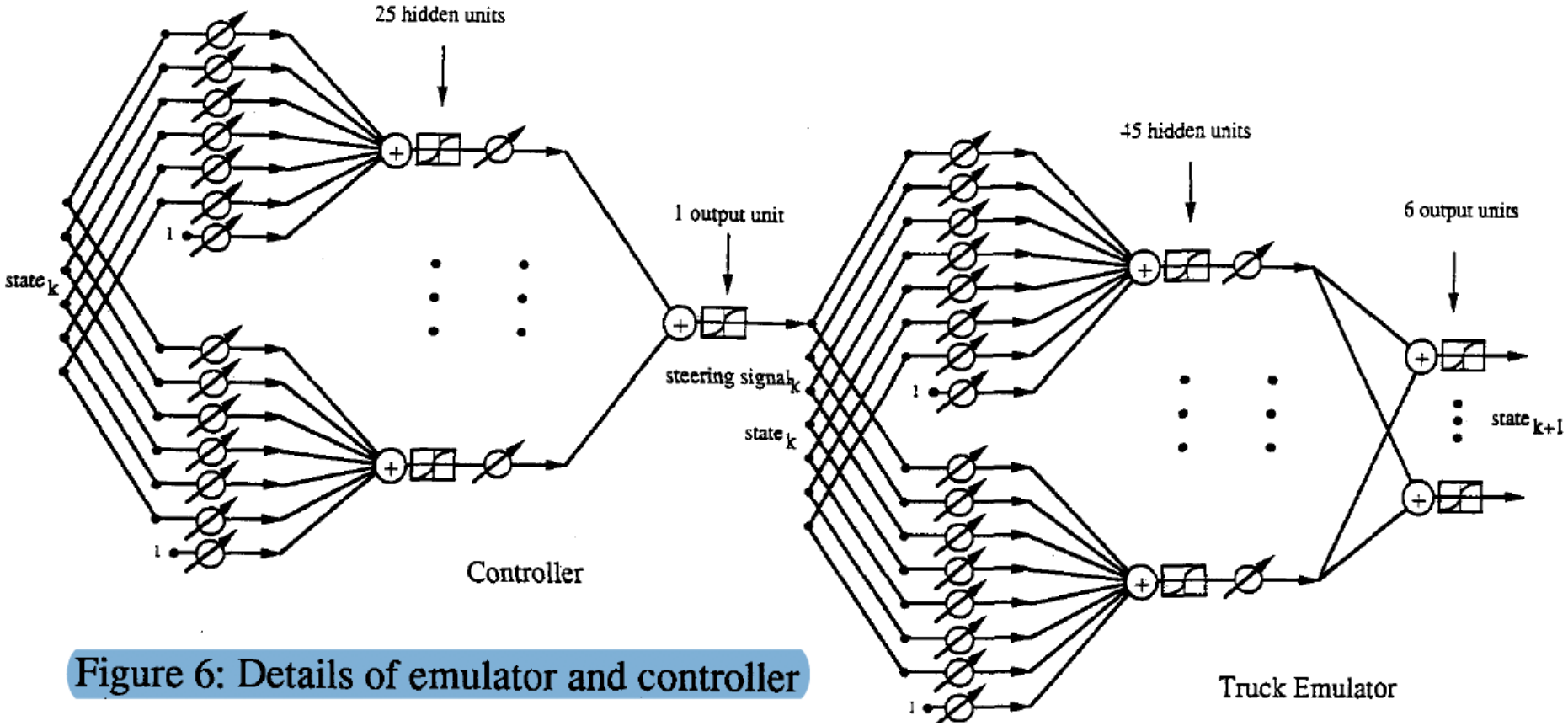
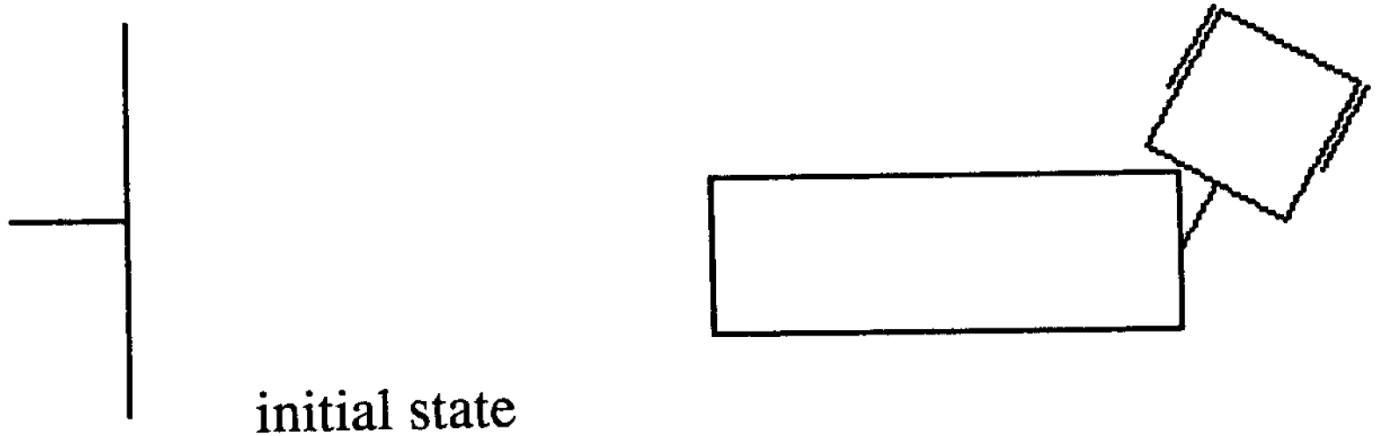
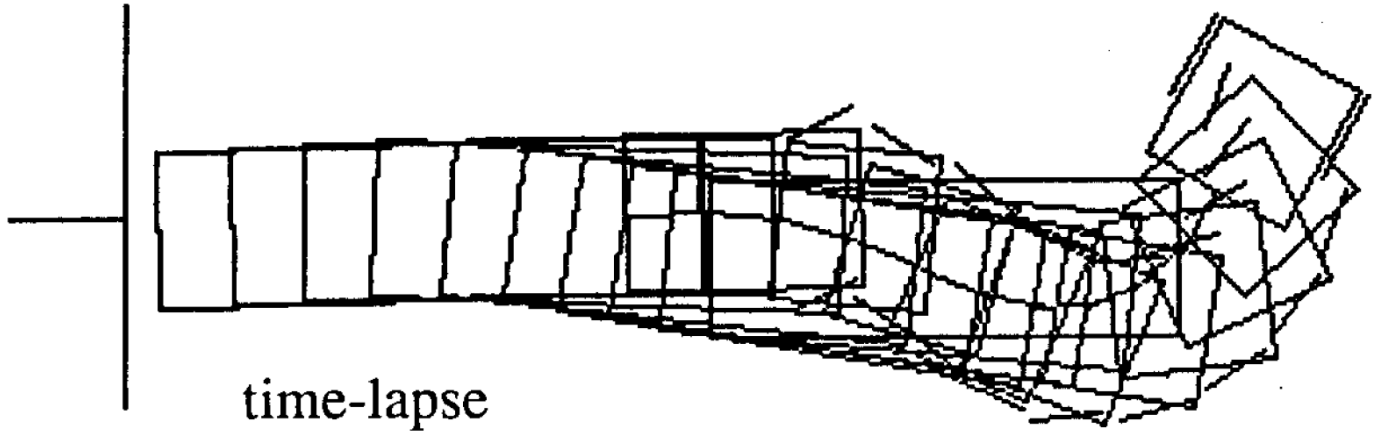


Figure 6: Details of emulator and controller

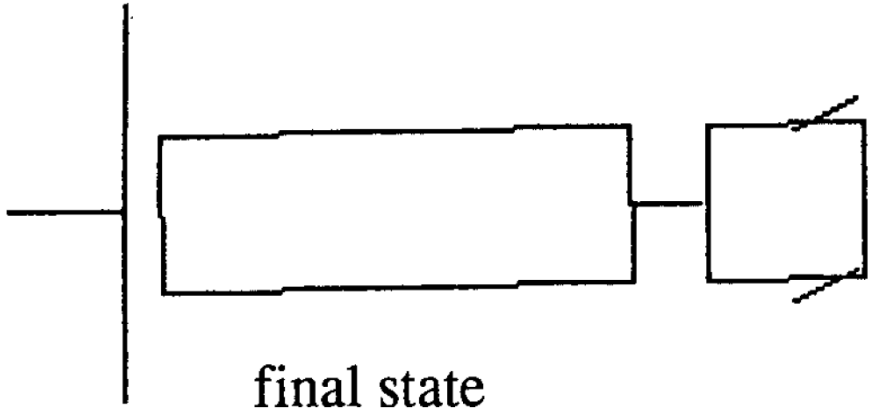
seen from Figures 4 and 5 to be analogous to a neural network having a number of layers equal to four times the number of backing up steps when going from the initial state to the final state. The number of steps varies of course with initial position of the truck and trailer.



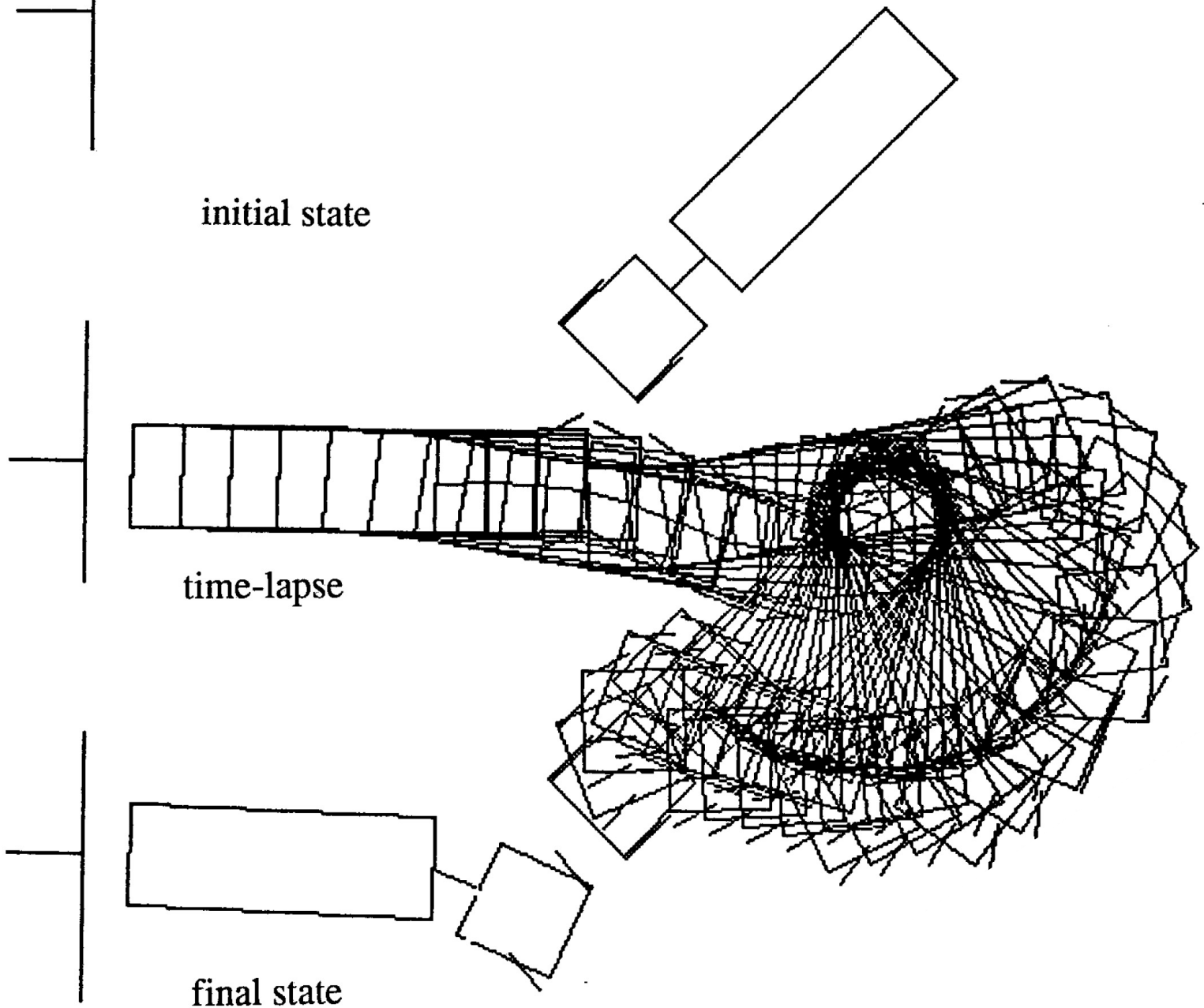
initial state

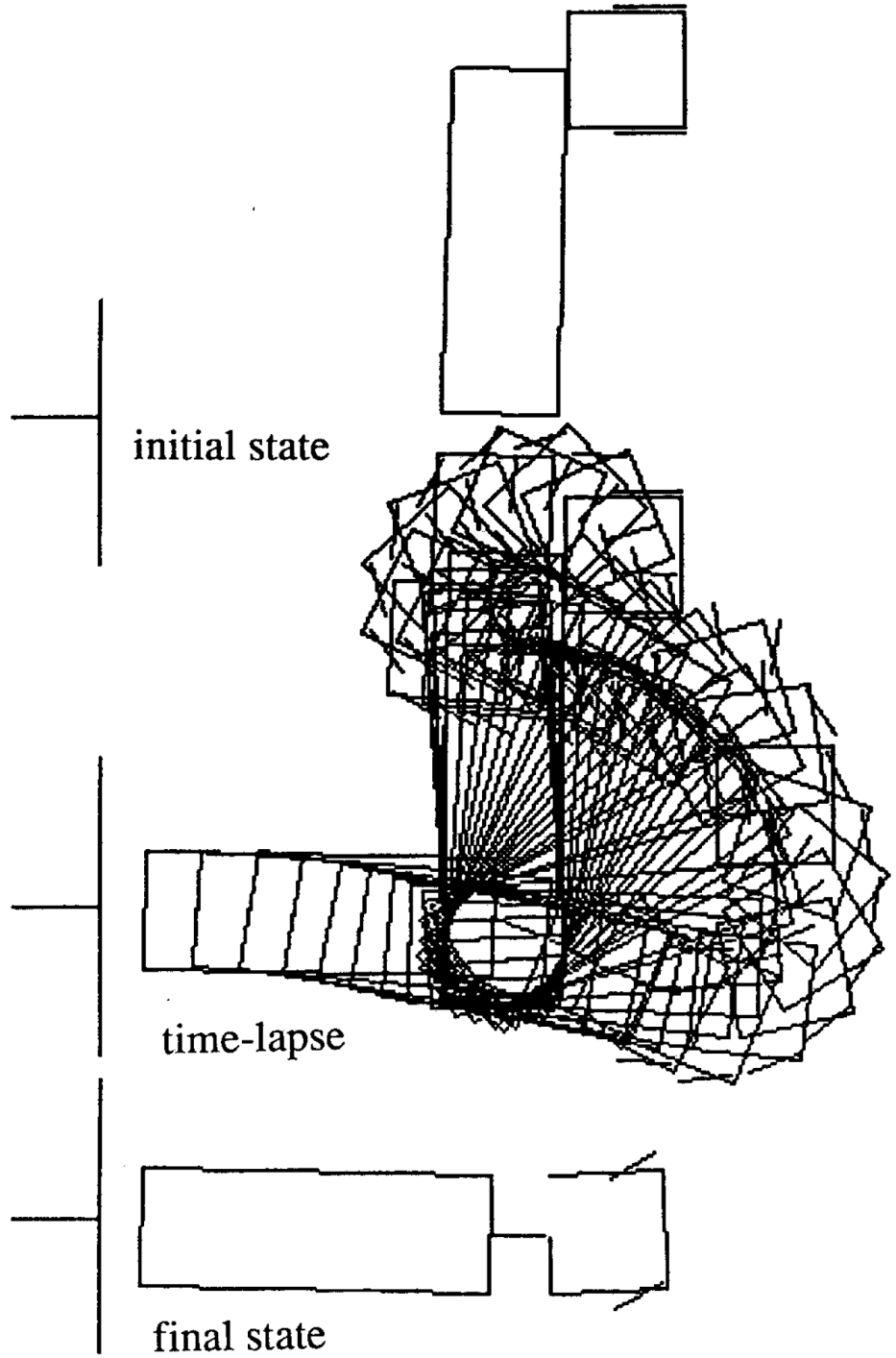


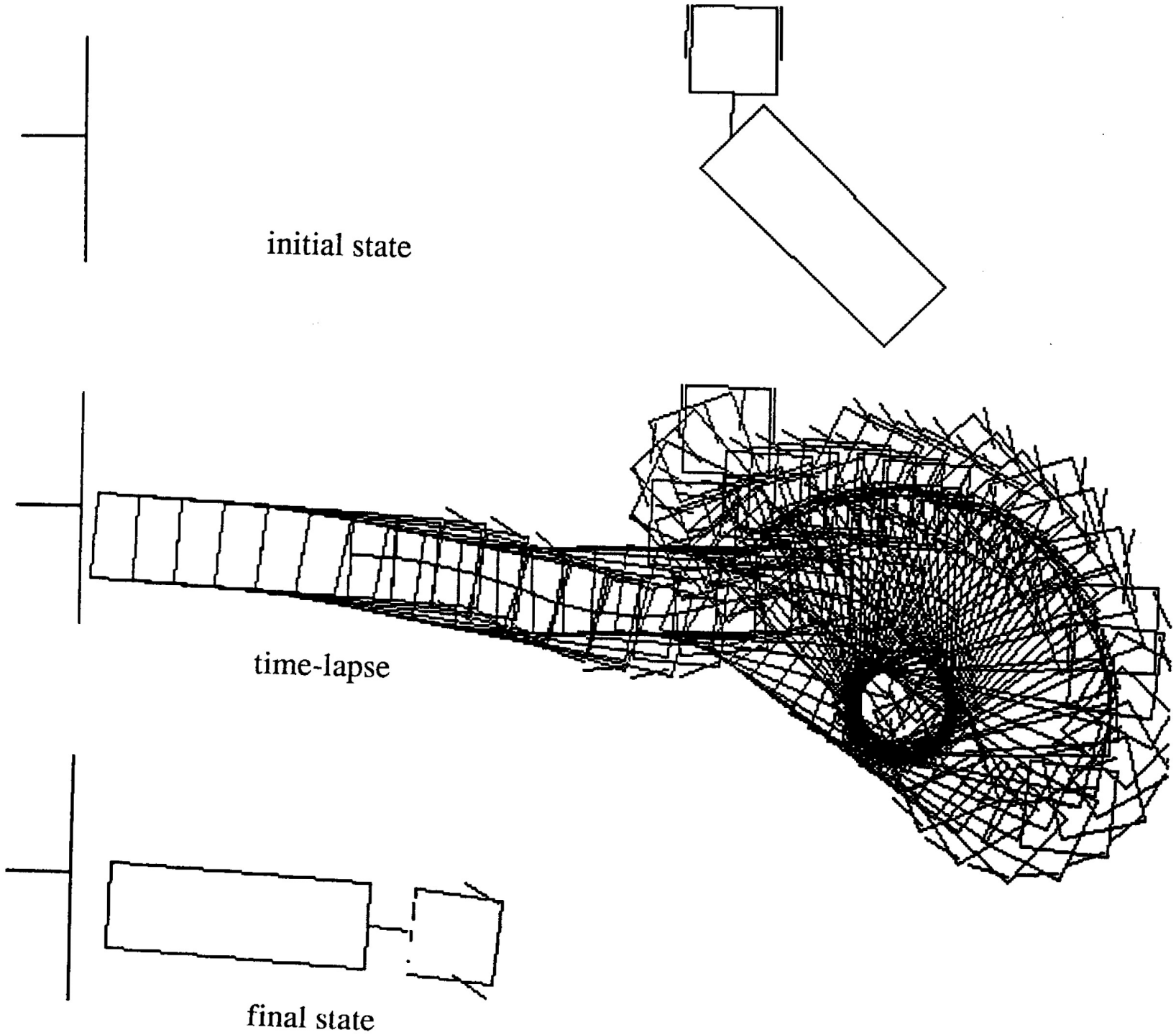
time-lapse



final state


initial state
time-lapse
final state


initial state
time-lapse
final state


initial state
time-lapse
final state

Additional resources

- Full working demo: https://tifu.github.io/truck_backer_upper/
- Nguyen & Widrow (1990) The truck backer-upper: An example of self-learning in neural networks
- Nguyen & Widrow (1990) Improving the learning speed of 2-layer neural networks by choosing initial values of the adaptive weights
- Jordan & Rumelhart (1992) Forward models: Supervised learning with a distal teacher
- Jenkins & Yuhas (1993) A simplified neural network solution through problem decomposition: The case of the truck backer-upper
- Schoenauer & Ronald (1994) Neuro-genetic truck backer-upper controller