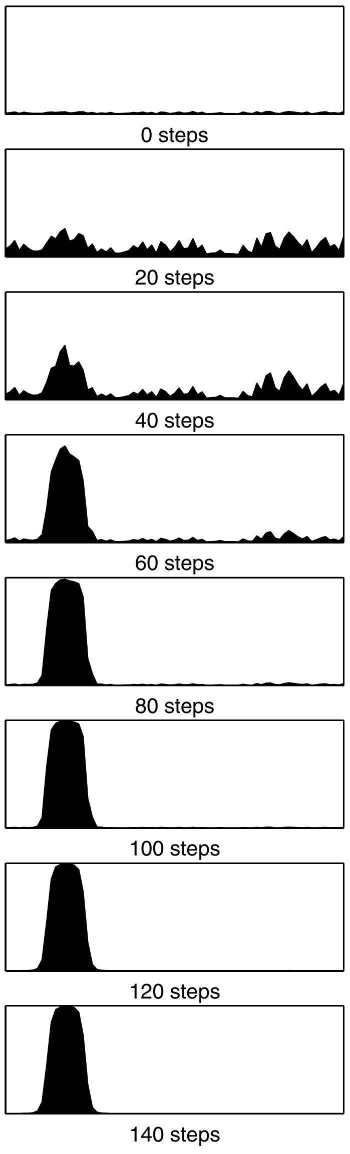
Appendix **A**: Attractor Networks

Attractor networks play important roles in the theories explaining the various components described in this book. There are two important cases: one- and two-dimensions. In the one-dimensional case, cells are assumed to be located along a ring (as head direction cells are; see chapter 5); and, in the two-dimensional case, to be located around a torus. This appendix will review only the one-dimensional case. The two-dimensional analogy is straightforward.

Imagine a population of cells. Let each cell be associated with a direction (0°–360°). If cells with nearby preferred directions are more strongly interconnected (i.e., have a higher synaptic weight) than cells with distant preferred directions and there is global inhibition, then this system will have dynamics such that a stable state is a hill of activation. No matter what state the system starts in, as long as there is some activity in the system and no external input, the system will settle to a single hill of activitation (see figure A.1). This local excitation, global inhibition connection structure has been studied by a number of researchers in one-dimension (Wilson and Cowan, 1973; Amari, 1977; Ermentrout and Cowan, 1979; Kishimoto and Amari, 1979; Kohonen, 1982, 1984; Skaggs et al., 1995; Redish, Elga, and Touretzky, 1996; Zhang, 1996; Redish and Touretzky, 1997b; Redish, 1997; Skaggs, 1997) and in two dimensions (Kohonen, 1982, 1984; Droulez and Berthoz, 1991; Munoz, Pélisson, and Guitton, 1991; Arai, Keller, and Edelman, 1994; McNaughton et al., 1996; Shen and McNaughton, 1996; Zhang, 1996; Redish and Touretzky, 1997b; Redish, 1997; Redish and Touretzky, 1998a; Samsonovich and McNaughton, 1997; Samsonovich, 1997).

Note that any direction is a stable attractor state and a coherent representation of direction. That is, the ring in figure 5.2 can



Put figure A.1b here.

Figure A.1
Simulation of an attractor network settling to a coherent representation from noise. Each panel (*left*) shows the population activity of the excitatory neurons in a simulation of a one-dimensional attractor. The activity of each neuron at each moment in time is plotted at its preferred direction. Three-dimensional plot of activity of each neuron by time (*above*). The left panels plot slices of the right taken at different times. Similar simulations have been reported by Kohonen (1982, 1984), Redish, Elga, and Touretzky (1996), Zhang (1996), Redish (1997), Samsonovich and McNaughton (1997), and Samsonovich (1997).

be rotated to any direction. Although there is no "global north," indeed no correspondence to north or south at all, the system knows that when the population peak is at the top of the ring, the animal is facing 180° from when the population peak is at the bottom of the ring. While the system can represent any direction and differentiate the different directions, the correspondence between represented direction and environment is not hard-wired — it must be inferred by observation.

EXTERNAL EXCITATORY INPUT

What is the effect of excitatory input onto this attractor network? There are four possible cases, depending on the location and magnitude of the extrapopulation input. (I assume input only synapses on excitatory neurons.)

1. If an attractor network is in a stable state and receives input (synapsing on excitatory cells) that is peaked at the same direction as is currently being represented, then nothing will change. The attractor network will still be in a stable state representing the same direction. The overall activity in the attractor network may change slightly, but the represented direction will not change.

2. On the other hand, if the input is offset slightly, then the attractor network will precess until the new representation is centered at the input direction (Skaggs et al., 1995; Redish, Elga, and Touretzky, 1996; Zhang, 1996; Samsonovich and McNaughton, 1997). figure A.2 shows an attractor network with offset input, which quickly precesses from its current representation to a new representation compatible with the input.

3. If input is weak and offset by a large angle, the current representation will not change (Redish and Touretzky, 1997b; Redish, 1997). figure A.3 shows an attractor network with offset input that does not change the current representation.

4. Finally, if input is strong enough and offset by a large angle, the representation of the current direction will disappear and the activity will reappear at the offset location (Zhang, 1996; Redish and Touretzky, 1997b; Redish, 1997; Samsonovich and McNaughton, 1997; see figure A.4).

The two simulations shown in figures A.2 and A.3 used the same input strengths. When weak excitatory drive is input near the peak of a coherent representation, the peak precesses to align itself with the offset drive. But if weak excitatory drive is offset substantially from the peak, then the peak does not move.

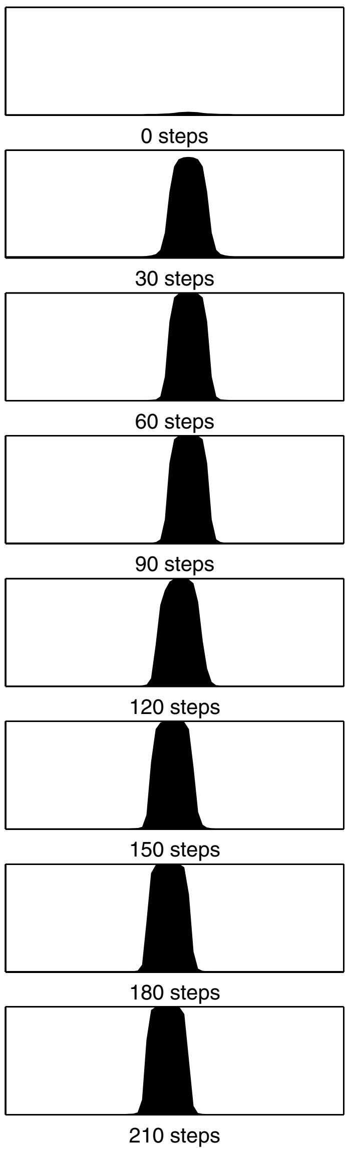
This makes a prediction in cue conflict situations where external cues conflict with internal representations. If the external cues suggest a candidate location near the currently represented location, the representation will shift to match the cues. If all candidates are distant from the current representation, the representation will not change. Near and distant can be quantitatively defined as twice the breadth of the tuning curve of the component cells.

Figure A.5 demonstrates this effect. It shows the final represented direction when a small extra excitation is input offset from the center of the bump. When the extra excitation is input near to the center of the bump, the bump precesses (figure A.2), but when the excitation is input far from the center, there is no effect (see also figure A.3). Note that the effect of the external input is linear out to approximately $\pm 60°$, but vanishes very quickly beyond that.

SETTLING FROM NOISE

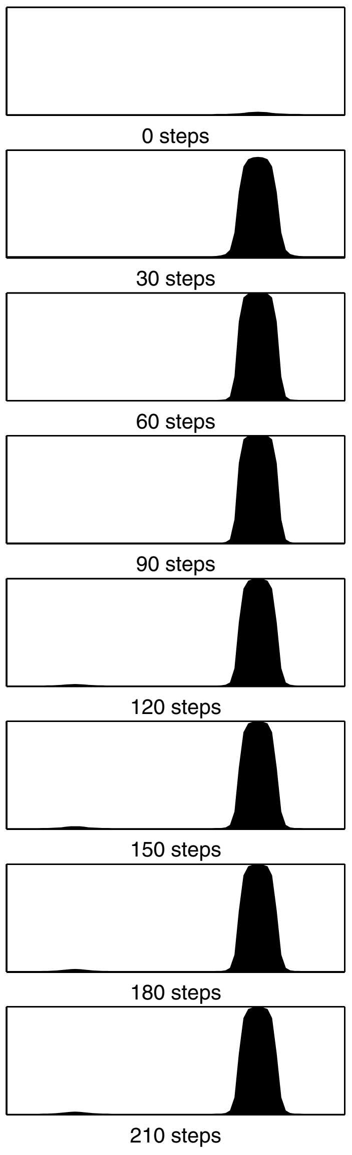
Figure A.1 shows a system settling from random noise. In chapter 9, it was argued that a two-dimensional attractor network settling from noise with biases provided by local view inputs could be used for self-localization. But what happens in cue-conflict situations where the biases do not all suggest the same location?

Even if an attractor network is initialized with a uniform random noise and two candidate biases are input the network will



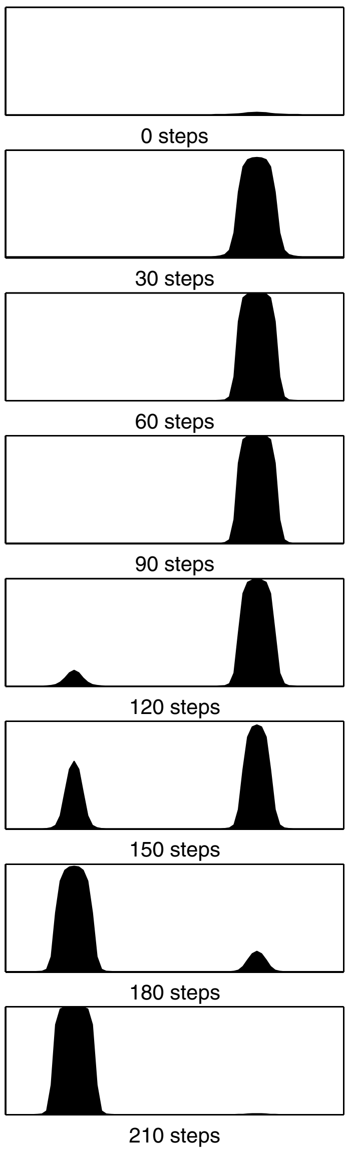
Put figure A.2b here.

Figure A.2
Simulation of an attractor network precessing from offset input. A small excitation is input slightly to the left of the center of the bump. Although this effect is small, continuously offset input is sufficient to track head direction accurately. (Compare figure A.4.) Similar simulations have been reported by Redish, Elga, and Touretzky (1996), Zhang (1996), Redish (1997), Samsonovich and McNaughton (1997), and Samsonovich (1997).



Put figure A.3b here.

Figure A.3
Simulation of an attractor network unaffected by offset input. A small excitation is input far from the center of the bump. (Compare figure A.4.) Similar simulations have been reported by Redish (1997).



Put figure A.4b here.

Figure A.4
Simulation of an attractor network with tonic excitatory input. A sufficiently large excitation is input far from the center of the bump. The activation bubble jumps from the old representation to the new one. (Compare figure A.2.) Similar simulations have been reported by Zhang (1996), Redish (1997), Samsonovich and McNaughton (1997), and Samsonovich (1997).

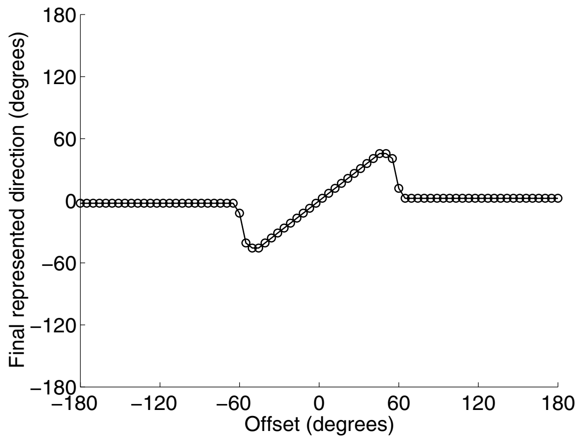
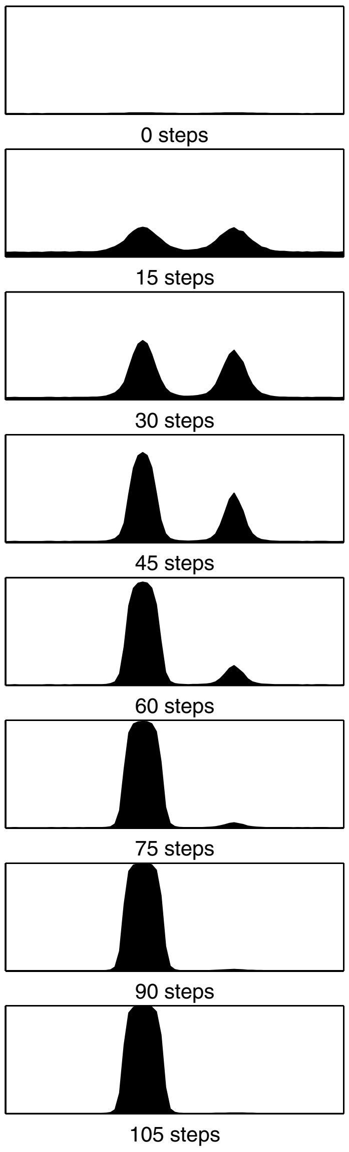


Figure A.5
Simulation of the influence of offset excitatory drive on final represented direction of an attractor network.

still settle to a single hill of activation. However, where this hill ends up depends on the separation of the locations of the two biases. If the biases are near each other, then the hill will be at the average of the two. If the biases are distant, then the hill will be at the location of one or the other. In other words, nearby biases average, but distal ones compete. This can be seen in figures A.6 and A.7.

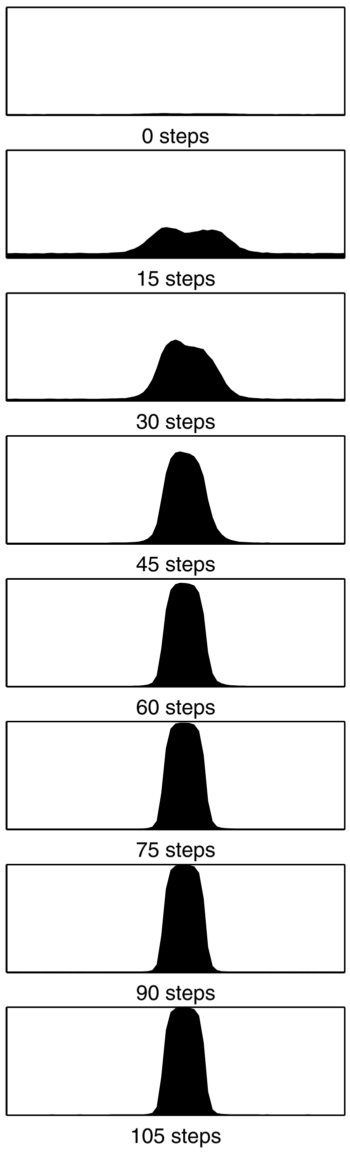
Although no one has yet shown averaging (the experiment has not been done), an example of competition has been reported by Collett, Cartwright, and Smith (1986) and replicated by Saksida et al. (1995, see also Redish, 1997), in which gerbils were trained to dig for food at a point halfway between two landmarks. When the gerbils were tested (without food) with the space between the landmarks doubled, they did not search at the center; instead, they searched at two locations, each the correct bearing and distance from one landmark.

This suggests an interesting experimental prediction, one that can be seen in either one dimension or two. For simplicity, I will present the prediction in one dimension. Train an animal to expect a small cue card (e.g. 5° width) in a standard cylindrical



Put figure A.6b here.

Figure A.6
Simulation of an attractor network settling from noise with two candidate biases having large separation. The two candidates compete because they are far apart. Compare figure A.7.



Put figure A.7b here.

Figure A.7
Simulation of an attractor network settling from noise with two nearby candidate biases. The two candidates merge because they are close together. Compare figure A.6. This dichotomy has been noted by Redish and Touretzky (1997b) and Skaggs (1997).

environment. Then, while recording from head direction cells (chapter 5), place the animal in the environment in the dark. Spin the environment quickly until the animal is dizzy. Then turn the lights on and allow the animal to see two cue cards separated by an angle θ. When θ is small, the head direction tuning curves should follow the average of the two cue cards, but when θ is large, the tuning curves should follow one or the other (the cards should compete). This prediction is a consequence of the hypothesis that the head direction system is an attractor network (see chapter 5).

EXAMPLE CODE

To complete this appendix, I include the `Matlab` 5.0 program used to generate the examples above. It simulates 75 excitatory neurons and one inhibitory interneuron using a neuronal model based on a variant of the Wilson-Cowan equations (Wilson and Cowan, 1973; see also Pinto et al., 1996). The figures in this appendix were generated by varying the `IN` array.

```
function RESULTS = Attractor(figure_to_show)
%
% Attractor
%
% One dimensional attractor sample code
% Written for Matlab 5.0
%
% Written by A. David Redish, 1998,
% for the book _Beyond the Cognitive Map_,
% published by MIT Press, 1999.

global dt
global W_EI W_IE W_II W_EE
global tauE gammaE tauI gammaI

%------------------------------
% set random seed
%------------------------------

rand('state', 0)
```

```
%-----------------------------
% PARAMETERS
%-----------------------------

NE = 75;         % Number of excitatory neurons

dt = 0.001;      % time step, in ms

tauE = 0.01;     % time constant, excitatory neurons
gammaE = -1.5;   % tonic inhibition, excitatory neurons

tauI = 0.002;    % time constant, inhibitory neuron
gammaI = -7.5;   % tonic inhibition, inhibitory neuron

E = zeros(NE,1); % allocate space for NE excitatory neurons
I = 0;           % one inhibitory neuron

W_EI = -8.0 * ones(NE,1);     % I -> E weights
W_IE = 0.880 * ones(1,NE);    % E -> I weights
W_II = -4.0;                  % I -> I weights
W_EE = BuildWeightMatrix(NE); % E -> E weights

%-----------------------------
% Input functions
%-----------------------------

switch figure_to_show

 case 1,             % figure A.1

  IN = cat(1, ones(100,1) * rand(1,NE), zeros(100,NE))';

 case 2,             % figure A.2

  IN0 = exp(-(((1:75) - 35).^2)/25);
  IN1 = ones(100,1) * IN0;
  IN0 = 2 * exp(-(((1:75) - 40).^2)/25);
  IN2 = ones(200,1) * IN0;
  IN = cat(1,IN1,IN2)';

 case 3,             % figure A.3

  IN0 = exp(-(((1:75) - 20).^2)/25);
  IN1 = ones(100,1) * IN0;
  IN0 = exp(-(((1:75) - 60).^2)/25);
  IN2 = ones(200,1) * IN0;
  IN = cat(1,IN1,IN2)';

 case 4,             % figure A.4
```

```
  IN0 = exp(-(((1:75) - 20).^2)/25);
  IN1 = ones(100,1) * IN0;
  IN0 = 2 * exp(-(((1:75) - 60).^2)/25);
  IN2 = ones(200,1) * IN0;
  IN = cat(1,IN1,IN2)';

 case 6,            % figure A.6

  IN0 = 0.5 * (0.1 * rand(1,75) + exp(-(((1:75) - 25).^2)/40));
  IN1 = 0.5 * (0.1 * rand(1,75) + exp(-(((1:75) - 45).^2)/40));
  IN = (ones(150,1) * (IN0 + IN1))';

 case 7,            % figure A.7

  IN0 = 0.5 * (0.1 * rand(1,75) + exp(-(((1:75) - 29).^2)/40));
  IN1 = 0.5 * (0.1 * rand(1,75) + exp(-(((1:75) - 41).^2)/40));
  IN = (ones(150,1) * (IN0 + IN1))';

 otherwise, disp('Unknown input set')

end

%----------------------------
% CYCLE
%----------------------------

steps = size(IN,2);

RESULTS = zeros(NE,steps);

for t=1:steps

 VE = W_EI * I + W_EE * E + gammaE + IN(:,t);
 VI = W_II * I + W_IE * E + gammaI;

 FE = 0.5 + 0.5 * tanh(VE);
 FI = 0.5 + 0.5 * tanh(VI);

 E = E + dt/tauE * (-E + FE);
 I = I + dt/tauI * (-I + FI);

 RESULTS(:,t) = E;

end
```

```
%-----------------------------
% DRAW
%------------------------------

figure(1)
clf
surfl(RESULTS);
shading interp
view([-30 75])
colormap(bone)
xlabel('time')
ylabel('neurons')
zlabel('activity')
set(gca,'Xtick',[])
set(gca,'Ytick',[])
set(gca,'Ztick',[])

figure(2)
clf
timeslices = 1:(steps/10):steps;
for i=1:8
 subplot(8,1,i);

 fill([1 1:NE NE],[0 RESULTS(:,timeslices(i))' 0],'k');
 axis([1 NE 0 1]);
 xlabel([num2str(timeslices(i)-1) ' steps'])
 set(gca,'Xtick',[]);
 set(gca,'Ytick',[]);

end

%------------------------------
% Build Weight Matrix
%------------------------------
function W = BuildWeightMatrix(NE)

stddevConst = 15;     % degrees
Wconst = 6.0;         % weight constant

variance = stddevConst^2 / (360^2) * NE ^2;

i = ones(NE,1) * (1:NE);
j = (1:NE)' * ones(1,NE);

d_choices = cat(3,abs(j + NE - i), abs(i + NE - j), abs(j - i));
d = min(d_choices, [], 3);

W = exp(-d .* d / variance);
W = Wconst * W./(ones(75,1) * sum(W));
```