A Study of Pierce's Group Generator

Thomas C. Henderson and Hongchang Peng
University of Utah

UUCS-10-004

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

1 December 2010

Abstract

Pierce describes an approach to map learning with uninterpreted sensors and effectors. As part of that, he describes a sensor grouping generator operator that attempts to arrange similar sensors into groups. Here we review that work and place it in a more strenuous statistical validation framework.

1 Introduction

Pierce [1] describes an approach to learning a model of the sensor set of an autonomous agent. Features are defined in terms of raw sensing data which exists as a specific set of types; e.g., scalar, vector, matrix, image element, image, field element, field and histogram. Feature operators are defined which map features to features, and the goal is to construct a perceptual system from this structure. The method used to accomplish this is to start with features from the raw sensory data, then generate new features and test their usefulness toward a goal.

One of the fundamental feature generators is the *grouping generator* which assigns features to a group if they are similar. This is based on the observation that similar sensors will produce similar features either at each instant (if neighboring and the world is mostly continuous) or over some sample period (if their histograms are similar over a reasonable sample period). Pierce's group generator functions by first defining metrics which capture the two aspects mentioned above (similar sample-wise or in their histograms), then determining subgroups which are similar in both metrics, and finally by taking the transitive closure of these subgroups.

In Chapter 4 of the dissertation, Pierce provides a simulation study to demonstrate the method. Our goal here is to repeat that study in order to duplicate the results and to explore various simulation issues in greater detail.

2 Pierce's Simulation Experiment

The simulation experiments are described in Chapter 4 of Pierce's dissertation. The first involves a mobile agent with a set of range sensors, a power level sensor, and four compass sensors. The sensors are grouped and then a structural layout in 2D is determined. The second experiment concerns an array of photoreceptors. Here we examine the first experiment, and in particular, the group generator.

2.1 Pierce's Experiment Definition

The basic setup involves a $6x4$ m^2 rectangular environment with a mobile robot defined as a point. The robot is equipped with 29 sensors all of which take values in the range from zero to one. Sensors 1 to 24 are range sensors which are arranged in an equi-spaced circle aiming outward from the robot. Although the dissertation states: "the sensors are numbered clockwise from the front," the structure given on p. 55 shows that they are numbered counter-clockwise; we also number them counter-clockwise since this is the positive direction of rotation in a right-handed coordinate frame with z coming up out of the plane. Range sensor 21 is defective and always returns the value 0.2. Sensor 25 gives the voltage level of the battery while sensors 26 to 29 give current compass headings for East, North, West and South, respectively. The value is 1 for the compass direction nearest the current heading and zero for the other compass sensors. There are two motors, a_0 and a_1, to drive the robot, and these can produce a maximum foward speed of 0.25 m/sec, and a maximum rotation speed of 100 degrees/sec. Although no details are given, we assume that the values of the motors range from -1 to 1, where -1 produces a backward motion and 1 produces a forward motion (more specifically, assume the rotational axis of the tracks is aligned with the y-axis; then a positive rotation moves z into x and corresponds to a positive rotation about y in the coordinate frame).

Some details of the motion model are left unspecified; therefore we use the following model:

```
if a0>= 0 and a1>=0
then robot moves forward min(a0,a1)*0.25 m/sec
     robot rotates ((a0-a1)/2)*100 degrees/sec

elseif a0<=0 and a1<=0
then robot moves backward abs(max(a0,a1))*0.25 m/sec
     robot rotates ((a0-a1)/2)*100 degrees/sec

elseif a0>0 and a1<0
then robot rotates ((a0-a1)/2)*100 degrees/sec

elseif a0>0 and a1<0
then robot rotates ((a0-a1)/2)*100 degrees/sec

end
```

Moreover, if the robot attempts to move out of the rectangular environment, no translation occurs, but rotation does take place.

Two metrics are defined:

$$d_{1,ij}(t) = \frac{1}{t+1} \sum_{\tau=0}^{t} \mid x_i(\tau) - x_j(\tau) \mid$$

where i and j are indexes of features i and j, $x_i(\tau)$ is the value of feature i at sample τ and t is some defined sample time.

$$d_{2,ij} = \frac{1}{2}(vsum(abs((pdfx_i) - (pdfx_j))))$$

where $pdfx_i$ is the histogram of x_i over all samples for sensor i, abs and $vsum$ are absolute value and vector sum as defined by Pierce.

Pierce runs the simulation for 5 simulated minutes and reports results on the sample data generated from that run. [Note that on p. 42, Pierce says: ”the robot wanders randomly for

2,500 steps," which is more like 4.17 minutes.] Based on the samples generated from this run, the group generator produces seven groups:

```
Range: {1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,
        22,23,24}
Defective range: {21}
Battery Voltage: {25}
Compass (East):  {26}
Compass (North): {27}
Compass (West):  {28}
Compass (South): {29}
```

Pierce then gives figures (Figures 4.3 and 4.4, p.53) showing the metrics d_1 and d_2 as surface plots (see Figures 1 and 2).

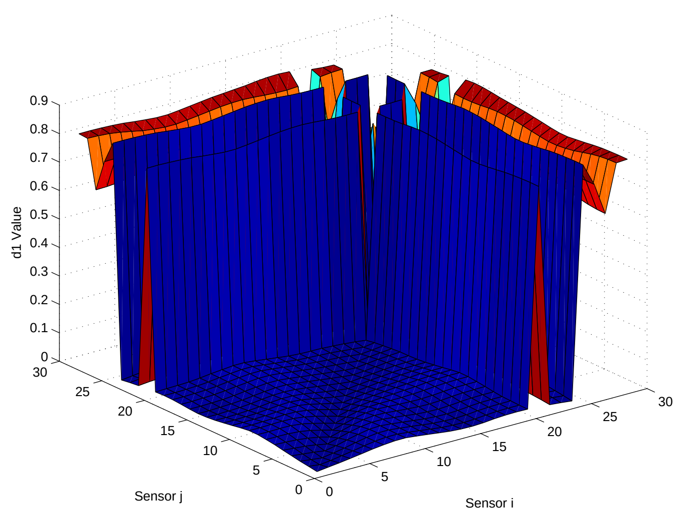

Figure 1: Pierce's d_1 Metric.

2.2 Discussion of Pierce's Experiment

Any simulation experiment should carefully state the questions to be answered by the experiment and attempt to set up a valid statistical framework. In addition, the sensitivity of the answer to essential parameters needs to be examined. Pierce does not explicitly formulate a question, nor name a value to be estimated, but it seems clear that some measure of

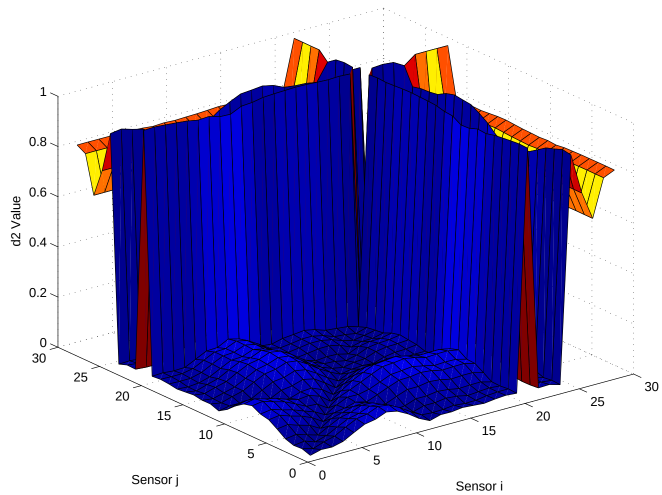

Figure 2: Pierce's d_2 Metric.

the correctness of the sensor grouping would be appropriate. From the description in the disertation, Pierce ran the experiment once for 5 minutes of simulated time, and obtained a perfect grouping solution.

From this we infer that the question to be answered is:

> **Grouping Correctness**: What is the correctness performance of the proposed grouping generator?

This requires a definition of correctness for performance and we propose the following:

Correctness Measure: Given (1) a set of sensors, $\{S_i, i = 1 : n\}$ (2) a correct grouping matrix, G, where G is an n by n binary valued matrix with $G(i,j) = 1$ if sensors S_i and S_j are in the same group and $G(i,j) = 0$ otherwise, and (3) H an n by n binary matrix which is the result of the grouping generator, then the grouping correctness measure is:

$$\mu_G(G,H) = \sum_{i=1}^{n}\sum_{j=1}^{n}(G(i,j) == H(i,j))/n^2$$

The major factors which influence the metrics and thus the grouping include:

1. *Environment*: the size of the environment, the number and placement of obstacles, the discontinuities, etc.; these all impact sensor values.

2. *Sensors and Effectors*: the variety of sensors, their placement, their range and noise characteristics all impact the sensed values.

3. *Algorithms:* the metrics used obviously influence the grouping, but their exploitation also has significant impact. For example, groupings could be made based on individual metrics and then combinations made on those groupings, or combinations of metrics can be used to produce the grouping (as is the case with Pierce's grouping generator). Also, any thresholds used in the algorithms play a direct role in the grouping. Certain parameters may also have an impact on the results; for example, the length of time selected to acquire data or even the number of bins used in a histogram.

Another significant issue is the set of assumptions made concerning the environment and the sensors. For example, are sample sensor values uniformly distributed given a set of uniformly distributed pose samples? In this experiment, this assumption holds for the compass sensors, but not for the range sensors (the max range value is 3 times more likely than any other value).

Finally, a statistical framework needs to be established in order to provide confidence in the results. Usually this means placing the estimates in a confidence interval determined from the variance of the individual estimates.

3 The Grouping Experiment Revisited

The set of questions of interest to us are:

1. How is grouping performance related to time (algorithmic)?
2. How is grouping performance related to the similarity threshold (algorithmic)?
3. How is grouping performance related to environment size (environment)?
4. How is grouping performance related to sensor noise, range and placement (sensors)?

In this section we discuss the questions, our approach, and the results.

3.1 Grouping Performance vs. Time

The robot starts in a random position and orientation in the environment (4x6 rectangle) and wanders about by setting random motor commands every second. Thus, the shorter the time it runs, the more biased the sensor samples will be by the initial pose. For example, if the robot starts at the center of the rectangle, then it takes at least four seconds to get closer than 1m to the boundary. Before that time, all non-defective range sensors will return a value of 1m. However, as time goes by, the range sensors should converge to the same histogram, and neighboring sensors should become more correlated. On the other hand, for runs below this amount of time, the range sensors should either all be very similar (initial pose in the center of the environment) or group according to whether the sensors face the boundary or not. For the compass sensors, their histograms should become more similar as time goes by, all converging to 25 % 1 values and 75 % 0 values, while their correlation value should approach 50 % (since half the time they are both 0 and equal, and the other half, one is 0 and the other 1).

3.1.1 Results

The method used here is to run trials of the robot for $1, 2, \ldots, 10$ minutes, performing 20 trials for each time selected. The results are shown in Figure 3. The error bars are for 95 % t-confidence intervals. The grouping correctness results here are significantly worse

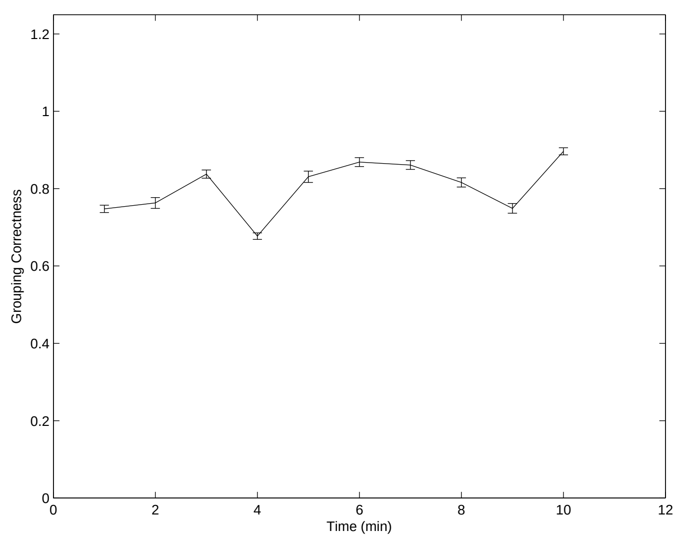

Figure 3: Grouping Correctness versus Time (95 % t-confidence intervals).

than Pierce's results (reported on 1 trial of 6000 steps). In investigating this further, we found the grouping to be highly dependent on the number of bins used in computing the d_2 distance measure. We discuss that now.

Impact of Number of Bins in Histogram Computation In the computation of the d_2 measure, the number of bins, n, used in the formation of the histogram impacts the result in the following way:

$$d_2 \rightarrow 1 \text{ as } n \rightarrow \infty$$

In the grouping generator, Pierce uses the minimum values of d_1 and d_2 to determine whether two sensors are grouped; thus, using a higher value of n results in a higher value of d_2, and this impacts the threshold (selected as the minimum of d_1 and d_2). Figure 4 shows the grouping performance with the number of bins set to 10 and set to 100. As can be seen, the performace is much better with the higher bin count. We assume therefore that Pierce used a higher bin count since he got perfect results.

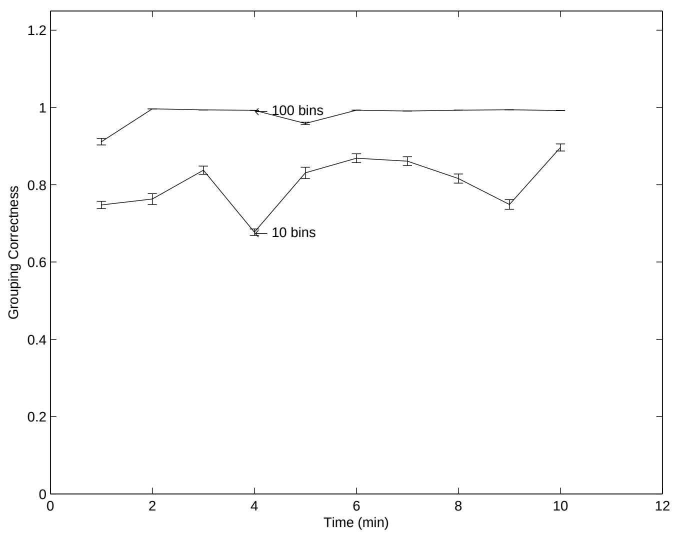

Figure 4: Grouping Correctness versus Time for Bin Values of 10 and 100.

In order to validate the intuitions about the Pierce metrics, we show the evolution of the d_1 metric for range sensors 1 and 2 in Figure 5, and for d_2 for the same 2 sensors in Figure 6. These are from a 10 minute run. Figures 7 and 8 show the evolution of metrics d_1 and d_2 for the East and North compass sensors for the same trial.

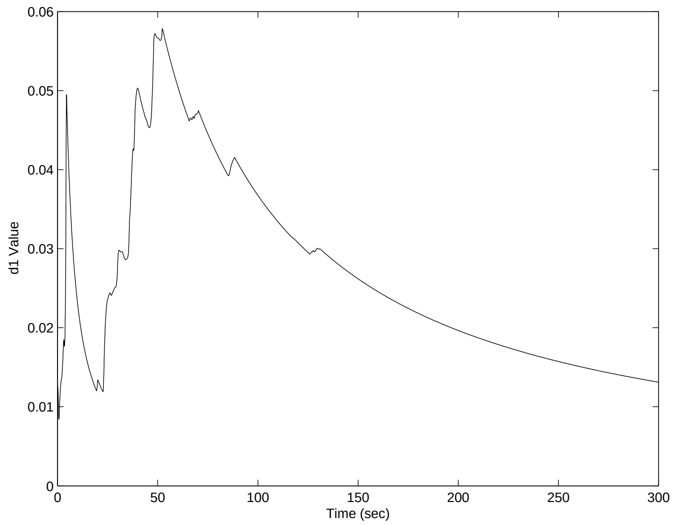

Figure 5: Evolution of d_1 Metric for Range Sensors 1 and 2.

3.2 Grouping Performance vs. Similarity Threshold

A major point of this study is to determine how the grouping threshold impacts performance. For example, does the change in performance as the threshold ranges from smaller to larger vary smoothly as well? The method used here is to run the robot for 20 minutes each trial and vary the threshold from the value of 1 to 20.5 in increments of 0.5. A total of 10 trials are run for each threshold value. Figure 9 shows the results.

The result indicates that as the grouping threshold increases from 1 to 3 (around here), there is a significant increase in the grouping correctness. From 3 to 8, the grouping performance is stable and has a value above 0.9. From 8 on, the grouping correctness drops gradually. All this indicates that the similarity threshold plays an important role in influencing the grouping performance. These results show that the optimum value for the grouping threshold is in the interval [4,8] instead of the value 2 used by Pierce.

3.3 Grouping Performance vs. Environment Size

Enlarging the environment should require a longer time to achieve high performance on grouping. This may be offset by the fact that as the environment size grows, the likelihood of ever getting close to the boundary goes down. Therefore, the performance behavior should start high, go down as boundaries affect the range sensors, and then go back up as

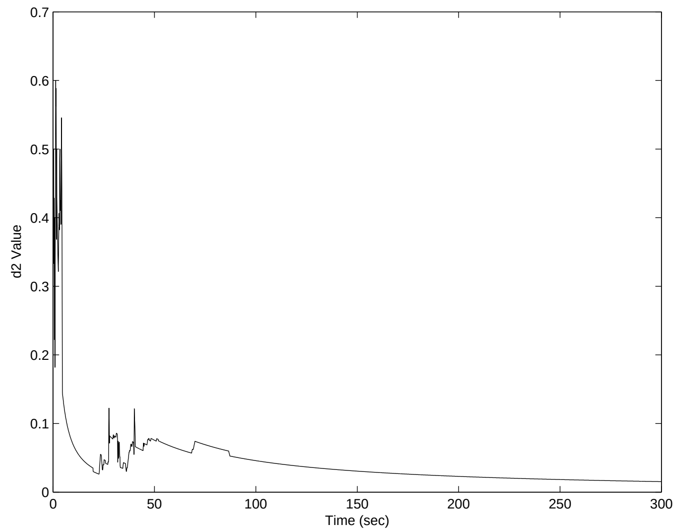

Figure 6: Evolution of d_2 Metric for Range Sensors 1 and 2.

more experience is gained. Of course, in an environment with a reasonably dense set of objects within sensor range, more varied range data would be accumulated more quickly.

3.3.1 Results

The method used here is to run each robot trial for 20 minutes, 10 sets each on environments of size 3x4, 4x6, and 6x8, thus doubling the environment area at each step. The performance results are shown in Figure 10.

This indicates that the grouping performance varies some as the environment size changes. However, the variance (indicated by the error bar) is also relatively large.

3.4 Grouping Performance vs. Noise

Noise effects will mainly impact the d_1 measure by de-correlating neighboring sensor values. Thus, the performance should get worse as the noise goes up.

The method used here is to run a 10 minute, 10 trials each with two types of noise: uni-

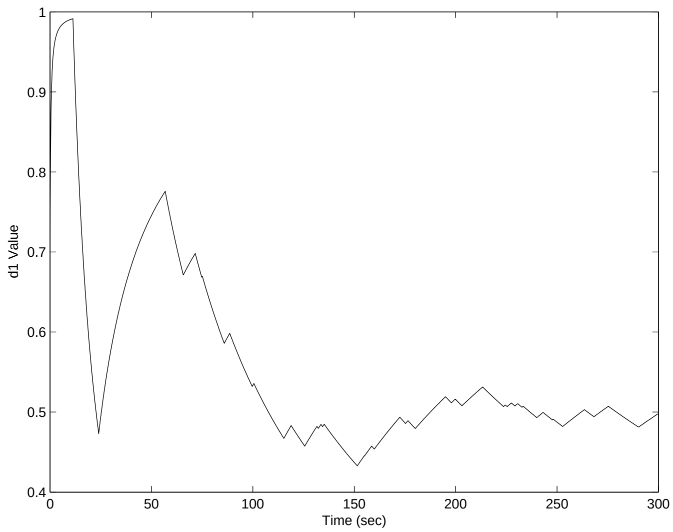

Figure 7: Evolution of d_1 Metric for East and North Compass Sensors.

formly distributed and normally distributed. For uniform noise, the value returned is:

$$v_U = v + U(-\beta, \beta)$$

where β is the maximum error allowed and $U(a, b)$ is a sample from the uniform distribution on the interval [a,b]. For Gaussian noise, the value returned is:

$$v_N = v + N(0, \sigma^2)$$

where σ^2 is the variance. Here we use $\beta \in \{0, 0.2, 0.4, 0.6, 0.8, 1\}$ and $\sigma^2 \in \{0, 0.001, 0.01, 0.1, 0.5, 1\}$ (in both cases 0 means no noise). Figures 11 and 12 show the results.

3.4.1 Uniformly Distributed Noise

As the noise level increases, the grouping performance increases. One possible explanation is that the randomness of the noise accompanied with the randomness of the sensor movement, has a positive effect on the grouping performance.

3.4.2 Normally Distributed Noise

This has shown that the normally distributed noise has a small negative impact on the grouping performance. The grouping correctness drops slightly for some variance values, but overall remains about the same.

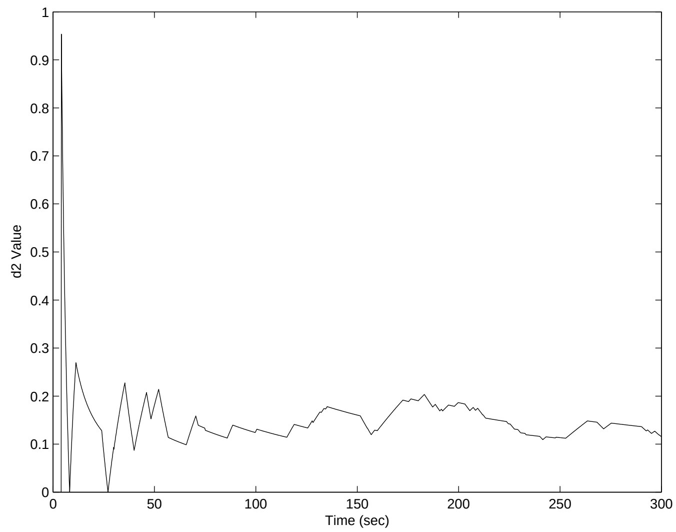

Figure 8: Evolution of d_2 Metric for East and North Compass Sensors.

3.5 Grouping Performance vs. Sensor Placement

The default setting is an equi-spaced circular placement of 24 range sensors. The potential influence of the sensor placement is explored here. Twelve range sensors are placed at the front and twelve at the back with one degree angle in between. The test evolves by time, from 2 minutes to 20 minutes with 2 minute gap in between, 10 trials for each time case. Figure 13 show the results. The figure shows that the grouping correctness fluctuates around 60% correctness, which is 25 % worse than the uniform circular placement. Thus, the effect of sensor placement has significant influence on the grouping performance.

3.6 Grouping Performance vs. Sensor Range

The range sensor in Pierce's experiment has a default maximum range of 1 meter. Here we allow the maximum range to vary from 1 to 9 meters. The tests are in a period of 20 minutes, doing 10 trials each for each maximum range case. Figure 14 shows the results.

It shows that the maximum range does not seem to influence performance. A maximum range of 3 meters yields a lower grouping correctness, but other than that, the results are stable

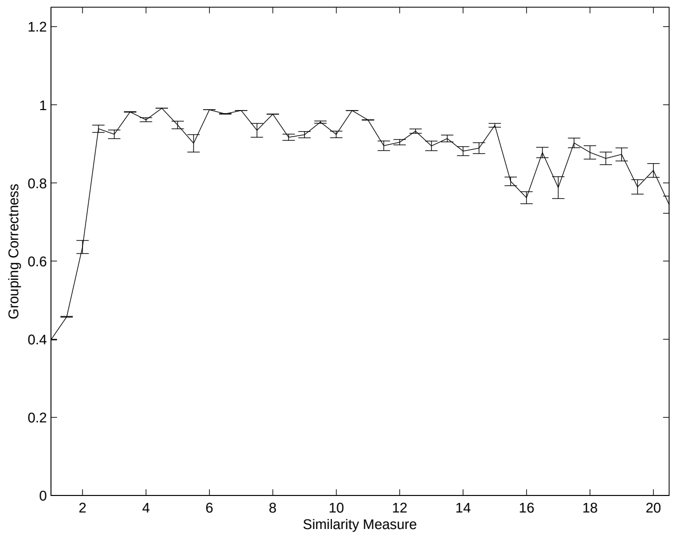

Figure 9: Grouping Performance vs Similarity Threshold.

4 Summary and Conclusions

This experiment has tested the performance of Pierce's algorithm in the default settings, and has also examined the role of bin size for the d_2 metric, the grouping threshold, environment size, sensor noise and placement, and maximum sensor range. The statistical analysis of Pierce's grouping generator has revealed the influential factors and the level of influence for each factor, thus leading to a better understanding of the grouping generator.

The bin sizes have turned out to be the most significant factor in influencing the grouping performance. In comparing 10 bins versus 100 bins, the results have an average margin of around 20% in grouping correctness. However, the choice of bin sizes has not been pointed out in Pierce's paper, and the automatic selection of bin size is an important issue in Pierce's framework. The grouping threshold, environment size, and sensor noise and maximum range also have significant impact on grouping performance. There exists a set of optimal values which yields the best performance, and it is still an open issue to find these. In addition, noise and sensor placement also influence grouping performance.

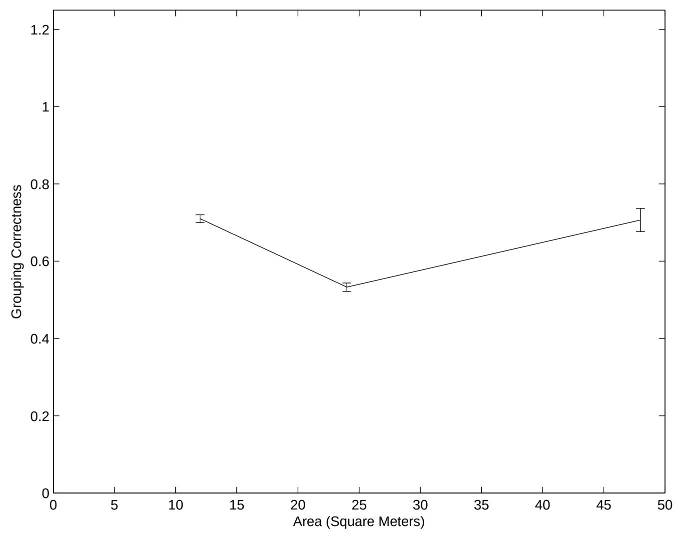

Figure 10: Grouping Performance vs. Environment Size.

5 Appendix A

```
function res = correct_TH(sg);
%
% correct.TH.m: This algorithm investigates the correctness
%     of a grouping result
% On Input:
%     sg: a 29*29 matrix
% On Output:
%     res: the correctness ratio
% Author:
%     T. Henderson
%     Univ of Utah
%     Oct 8th, 2010
%

RANGE_GROUP = [1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,0,1,1,1,0,0,0,0,0];
DEF_GROUP =   [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0,0,0,0,0];
BAT_GROUP =   [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0,0];
EAST_GROUP =  [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0,0];
NORTH_GROUP = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0,0];
WEST_GROUP =  [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1,0];
SOUTH_GROUP = [0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,1];
tot_max = 29ˆ2;

tot = 0;
for r = 1:20
   tot = tot + sum(sg(r,:)==RANGE_GROUP);
end
tot = tot + sum(sg(21,:)==DEF_GROUP);
for r = 22:24
   tot = tot + sum(sg(r,:)==RANGE_GROUP);
end
tot = tot + sum(sg(25,:)==BAT_GROUP);
```

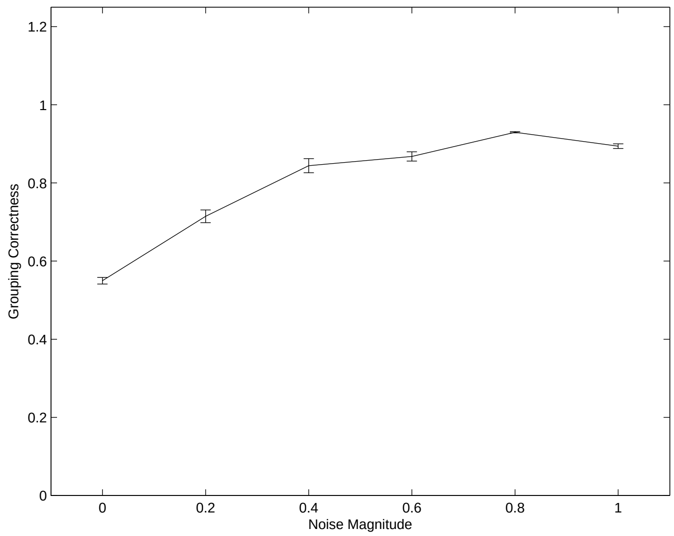

Figure 11: Grouping Performance with Uniformly Distributed Noise.

```
tot = tot + sum(sg(26,:)==EAST_GROUP);
tot = tot + sum(sg(27,:)==NORTH_GROUP);
tot = tot + sum(sg(28,:)==WEST_GROUP);
tot = tot + sum(sg(29,:)==SOUTH_GROUP);
res = tot/tot_max;



function [d1,d2] = distance_metric(sm,col);
%
% On Input:
%     sm:sensory input matrix
% On Output:
%    d1,d2: 29*29 sensor correlation matrix
%

% get time and number of sensors
sm = sm(:,1:29);
[t,n_s] = size(sm);
d1 = zeros(n_s,n_s);
d2 = zeros(n_s,n_s);

pdfi = zeros(n_s,col);
for ind = 1:n_s
    pdfi(ind,:) = hist(sm(:,ind),col);
    pdfi(ind,:) = pdfi(ind,:)/sum(pdfi(ind,:));
end

for i = 1:n_s
    for j = 1:n_s
        d1(i,j) = sum(abs(sm(:,i)-sm(:,j)))/t;
%         if d1(i,j)>max_range
%             disp([i j d1(i,j)]);
%             pause;
%         end
        d2(i,j) = sum(abs(pdfi(i,:)-pdfi(j,:)))/2;
```

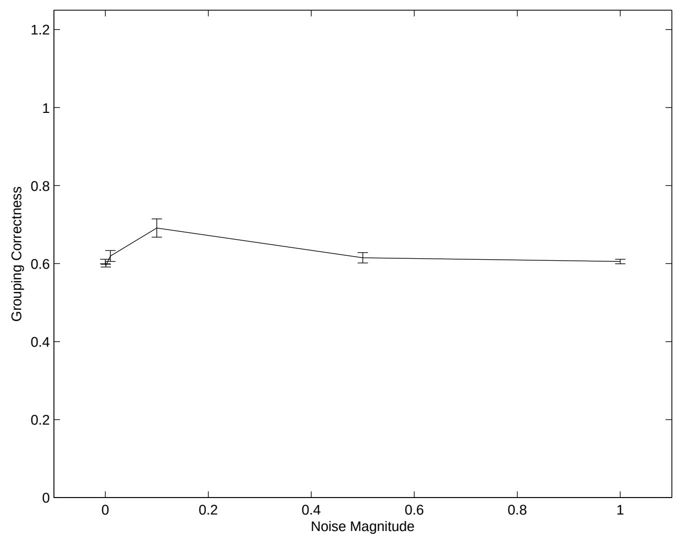

Figure 12: Grouping Performance with Normally Distributed Noise.

```
    end
end




function s = PIE_29sensors_2(x,y,theta,t,motors,max_range,...
                             x_max,y_max);
%
% PIE_29sensors_2.m: getting the sensory signals of 29 sensors
% On Input:
%     (x,y): current location
%     angle: forward angle
%     t: time has consumed since beginning
% On Output:
%     s: 29*1 vector storing the values of sensory signals
% Author:
%     H. Peng
%     Univ of Utah
%     Sep 8th, 2010
%

num = 34;
s = zeros(num,1);
unitAngle = 2*pi/24;
for ind = 1:24
    angle = theta + unitAngle*(ind-1);
    x1 = 0;  y1 = y - x*tan(angle);
    x2 = x_max;  y2 = y + x_max*tan(angle) - x*tan(angle);
    x3 = -1/tan(angle)*(y - x*tan(angle)); y3 = 0;
    x4 = 1/tan(angle)*(x*tan(angle) - y + y_max); y4 = y_max;

    dist = -1;
    if angle == pi/2
        dist = y_max - y;
```

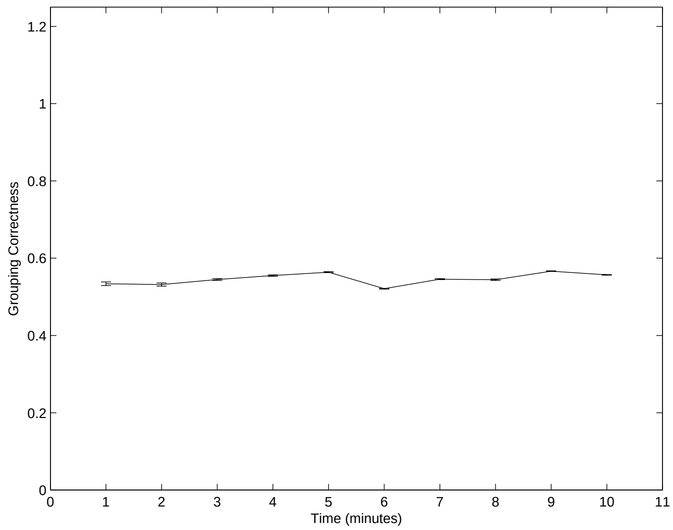

Figure 13: Grouping Performance with Modified Sensor Placement.

```
    elseif angle == pi/2*3;
        dist = y;
    elseif angle == 0
        dist = x_max - x;
    elseif angle == pi
        dist = x;
    else
        if y1>=0 && y1<=y_max && (y1-y)/sin(angle)>=0
            dist = sqrt((x-x1)^2+(y-y1)^2);
        elseif y2>=0 && y2<=y_max && (y2-y)/sin(angle)>=0
            dist = sqrt((x-x2)^2+(y-y2)^2);
        elseif x3>=0 && x3<=x_max && (x3-x)/cos(angle)>=0
            dist = sqrt((x-x3)^2+(y-y3)^2);
        elseif x4>=0 && x4<=x_max && (x4-x)/cos(angle)>=0
            dist = sqrt((x-x4)^2+(y-y4)^2);
        end
    end

    % the default max_range in the paper is 1 meter
    if dist>=max_range
        dist = max_range;
    end
    s(ind,1) = dist;
end

s(21,1) = 0.2;
s(25,1) = 1 - t/100000;
s(26:29,1) = 0;
t = theta/pi*4;
t = mod(t,8);

if t<=1 || t>7
    s(26,1) = 1;%E
elseif t>1 && t<=3
    s(27,1) = 1;%N
elseif t>3 && t<=5
```

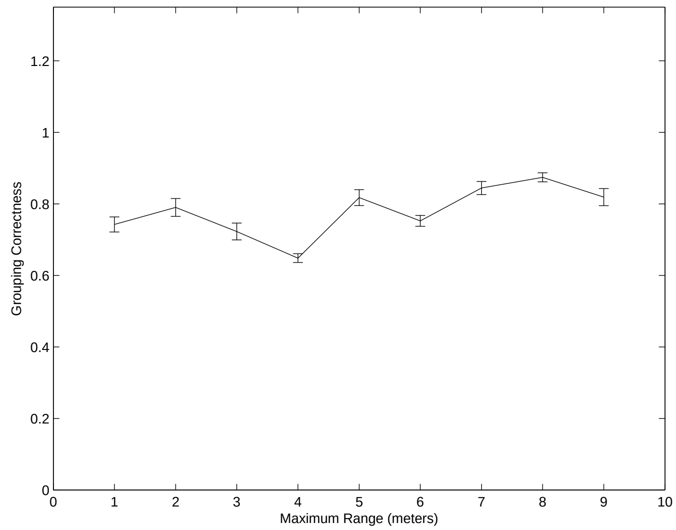

Figure 14: Grouping Performance of Different Maximum Sensor Range.

```
    s(28,1) = 1;%W
else
    s(29,1) = 1;%S
end

s(30,1) = x;
s(31,1) = y;
s(32,1) = theta;
s(33,1) = motors(1);
s(34,1) = motors(2);




function [group,belong2] = PIE_cluster(related);
%
% PIE_cluster.m: classified how many correct grouping
%     are there inside the related matrix
% On Input:
%     related: 29*29 matrix
% On Output:
%     group: a cell array, each cell contains the related sensors
%     belong2: a 29*1 vector showing which cell each sensor
%              belongs to
% Author:
%     H. Peng
%     Univ of Utah
%     Sep 8th, 2010
%

group = {};
% see each sensor vector belongs to which group
belong2 = zeros(29,1);
label = 0;

for ind = 1:29
```

```
    if belong2(ind) == 0
        label = label + 1;
        group{label} = ind;
        belong2(ind) = label;
        for jnd = ind+1:29
            if related(ind,jnd) == 1 && check(group{label},jnd)==0
                group{label} = [group{label} jnd];
                belong2(jnd) = label;
            end
        end
    end
end

function resu = check(Arr,b);
% check the existence of some value in an array
for ind = 1:length(Arr)
    if Arr(ind) == b
        resu = 1;
    end
end
resu = 0;
```

```
function [xp,yp,thetap] = PIE_move_TH(x,y,theta,motors,...
                                      x_max,y_max)
%
% PIE_move - advance robot according to motor values
% On input:
%     x (float): x location (0<=x<=4)
%     y (float): y location (0<=y<=6)
%     theta (float): direction
%     motors (1x2 vector):
%       element 1: a0 value  (-1<=a0<=1)
%       element 2: a1 value  (-1<=a1<=1)
% On output:
%     xp (float): new x position (0<=xp<=3)
%     yp (float): new y position (0<=yp<=6)
%     thetap (float): new thetap angle
%     Note: if motion leaves 3x6 rectangle, then new heading
%           is old heading plus 180 degrees
% Call:
%     [x,y,theta] = PIE_move(2,3,0,[0.5,0.5]);
%     Should return [3,2.025,0]
% Author:
%     T. Henderson
%     UU
%     Fall 2010
%
index = 1;

MAX_VEL = 0.025;
MAX_ROT = 10*pi/180;
ZERO_EPS = 0.001;
% x_max = 4;
% y_max = 6;
a0 = motors(1);
a1 = motors(2);

m_max = max(a0,a1);
```

```
m_min = min(a0,a1);
if a0>=0&&a1>=0
  deld = m_min*MAX_VEL;
  rotd = ((m_max-m_min)/2)*MAX_ROT;
  if a1>a0
      rotd = -rotd;
  end
elseif a0<=0&&a1<=0
  deld = max(a0,a1)*MAX_VEL;
  rotd = ((m_max-m_min)/2)*MAX_ROT;
  if a1>a0
      rotd = -rotd;
  end
elseif a0<0&&a1>0
  deld = 0;
  rotd = -((abs(a0)+abs(a1))/2)*MAX_ROT;
else
  deld = 0;
  rotd = ((abs(a0)+abs(a1))/2)*MAX_ROT;
end
xd = deld*cos(theta) * index;
yd = deld*sin(theta) * index;
xp = x + xd;
yp = y + yd;
thetap = posori(theta + rotd);
if xp>x_max-ZERO_EPS
  xp = x_max-ZERO_EPS;
elseif xp<=ZERO_EPS
  xp = ZERO_EPS;
end
if yp>y_max-ZERO_EPS
  yp = y_max-ZERO_EPS;
elseif yp<=ZERO_EPS
  yp = ZERO_EPS;
end
```

```
function s_m = PIE_robot(x,y,theta,motors,num_min,...
                          max_range,x_max,y_max);
%
% PIE_robot2: random move and collects the signals
% On Input:
%     x,y: the starting location
%     theta: the starting angle
%     motors: the N*2 motor vector
%     num_min: number of minutes for random movement
%     max_range: the setting of maximum range for range sensors
%     x_max,y_max: the length and width of the environment
% On Output:
%     s_m:(10*num_min,33) matrix of sensor signals
% Author:
%     H. Peng
%     Univ of Utah
%     Sep, 2010
%

s_m = zeros(10*60*num_min,34);
seconds = 0;
while(seconds < 60*num_min)
```

```
    for ind = 1:10
        s = PIE_29sensors_2(x,y,theta,seconds+0.1*ind-0.1,...
                motors(10*seconds+ind,:),max_range,x_max,y_max);
        s_m(10*seconds+ind,:) = s';
        [xp,yp,thetap] = PIE_move_TH(x,y,theta,...
                      motors(10*seconds+ind,:),x_max,y_max);
        x = xp; y = yp; theta = thetap;
    end
    seconds = seconds + 1;
end



function [s_m] = PIE_robot_noisy_gaussian(x,y,theta,motors,...
                      num_min,max_range,x_max,y_max);
%
% PIE_robot_noisy_gaussian: random move and collects the signals,
%     and also the gaussian noise is inserted to the signals
% On Input:
%     x,y: the starting location
%     theta: the starting angle
%     motors: the N*2 motor vector
%     num_min: number of minutes for random movement
%     max_range: the setting of maximum range for the range sensors
%     x_max,y_max: the length and width of the environment
% On Output:
%     s_m: a 6*1 cell array, each cell is a (10*num_min,33) matrix of
%          sensor signals
% Author:
%     H. Peng
%     Univ of Utah
%     Sep, 2010
%


s_m = cell(6,1);
for i = 1:6
    s_m{i} = zeros(10*60*num_min,34);
end

seconds = 0;
while(seconds < 60*num_min)
    for ind = 1:10
        s = PIE_29sensors_2(x,y,theta,seconds+0.1*ind-0.1,...
                   motors(10*seconds+ind,:),max_range,x_max,y_max);
        for i = 1:6
            s_m{i}(10*seconds+ind,:) = s';
        end
        for k = 2:6
            for jnd = 1:29
             x = s_m{1}(10*seconds+ind,jnd);
             s_m{2}(10*seconds+ind,jnd) = s_m{k}(10*seconds+ind,jnd)...
                           + 1/sqrt(2*pi*0.001)*exp(-1/2*x^2/0.001);
             s_m{3}(10*seconds+ind,jnd) = s_m{k}(10*seconds+ind,jnd)...
                             + 1/sqrt(2*pi*0.01)*exp(-1/2*x^2/0.01);
             s_m{4}(10*seconds+ind,jnd) = s_m{k}(10*seconds+ind,jnd)...
                               + 1/sqrt(2*pi*0.1)*exp(-1/2*x^2/0.1);
             s_m{5}(10*seconds+ind,jnd) = s_m{k}(10*seconds+ind,jnd)...
                               + 1/sqrt(2*pi*0.5)*exp(-1/2*x^2/0.5);
             s_m{6}(10*seconds+ind,jnd) = s_m{k}(10*seconds+ind,jnd)...
                               + 1/sqrt(2*pi*1)*exp(-1/2*x^2/1);
```

```
            end
        end
        [xp,yp,thetap] = PIE_move_TH(x,y,theta,...
                                    motors(10*seconds+ind,:),x_max,y_max);
        x = xp; y = yp; theta = thetap;
    end
    seconds = seconds + 1;
end
```

```
function [s_m] = PIE_robot_noisy_uniform(x,y,theta,motors,...
                                            num_min,max_range,x_max,y_max);
%
% PIE_robot2_noisy_uniform: random move and collects the signals, also
%     the uniform noise is inserted to the signals
% On Input:
%     x,y: the starting location
%     theta: the starting angle
%     motors: the N*2 motor vector
%     num_min: number of minutes for random movement
%     max_range: the setting of maximum range for the range sensors
%     x_max,y_max: the length and width of the environment
% On Output:
%     s_m: a cell array, each cell is a (10*num_min,33) matrix of
%          sensor signals
% Author:
%     H. Peng
%     Univ of Utah
%     Sep, 2010
%

s_m = cell(6,1);
for i = 1:6
    s_m{i} = zeros(10*60*num_min,34);
end
%

seconds = 0;
while(seconds < 60*num_min)
    for ind = 1:10
        s = PIE_29sensors_2(x,y,theta,seconds+0.1*ind-0.1,...
                         motors(10*seconds+ind,:),max_range,x_max,y_max);
        for i = 1:6
            s_m{i}(10*seconds+ind,:) = s';
        end
        for k = 2:6
            for jnd = 1:29
                s_m{k}(10*seconds+ind,jnd) = s_m{k}(10*seconds+ind,jnd)...
                        + 0.2*(k-1)*(-1+2*rand());
            end
        end
        [xp,yp,thetap] = PIE_move_TH(x,y,theta,motors(10*seconds+ind,:),...
                        x_max,y_max);
        x = xp; y = yp; theta = thetap;
    end
    seconds = seconds + 1;
end
```

```
function s = PIE_run_robot(x0,y0,theta0,motors,max_range)
%
% PIE_run_robot - run robot for initial state and motor values given
% On input:
%     x0 (float): initial x location (0<=x<=4)
%     y0 (float): initial y location (0<=y<=6)
%     theta0 (float): initial direction
%     motors (nx2 array): motor values to apply [a0,a1]
%                         applied every 0.1 sec
%     max_range (float): return this value if range >= to it
% On output:
%     s (nx34 array): sensor and state values each time step
%       indexes 1-24: range values at 24 equi-spaced angles (first
%                     value is straight ahead
%                     index 20 returns constant value of 0.2
%       index 25: battery level
%       index 26: Compass value: East
%       index 27: Compass value: North
%       index 28: Compass value: West
%       index 29: Compass value: South
% Call:
%     s = PIE_run_robot(2,3,0,ones(10,2),1);
% Author:
%     T. Henderson
%     UU
%     Fall 2010
%

MAX_JUMP = 0.5;
NUM_SENSORS = 29;
NUM_STATE_VARS = 5;

x_max = 4;
y_max = 6;

x = x0;
y = y0;
theta = theta0;
[num_steps,~] = size(motors);
s = zeros(num_steps+1,NUM_SENSORS+NUM_STATE_VARS);
t = 0;
% notice that #21 sensor is not defective yet!!!!
samp = PIE_29sensors_2(x,y,theta,t,zeros(1,2),max_range,4,6);
samp = samp';
s(1,1:29) = samp(1,1:29);
s(1,30) = x;
s(1,31) = y;
s(1,32) = theta;
s(1,21) = 0.2;

for step = 1:num_steps
  motors_now = motors(step,:);
  samp = PIE_29sensors_2(x,y,theta,t,motors(step,:),max_range,4,6);
  samp = samp';
  s(step+1,1:29) = samp(1,1:29);
  [x,y,new_theta] = PIE_move_TH(x,y,theta,motors_now,4,6);
  new_theta = posori(new_theta);
%   if abs(new_theta-theta)>MAX_JUMP  % if 180 degree direction change
%       tmp1 = s(step+1,21);
%       s(step+1,1:12) = samp(13:24);
%       s(step+1,13:24) = samp(1:12);
%       s(step+1,9) = tmp1;
```

```
%   end
  s(step+1,21) = 0.2;
  theta = new_theta;
  s(step+1,30) = x;
  s(step+1,31) = y;
  s(step+1,32) = theta;
  s(step+1,33) = motors_now(1);
  s(step+1,34) = motors_now(2);
end



function [related,sim1,sim2] = PIE_subGroup2(d1,d2,grouping_threshold);
%
% use distance metrics to form subgroups of similar sensors
% this is the original version of the author without any modification
% Author:
%     H. Peng
%     Univ of Utah
%     Sep, 2010
%

[n_s ~] = size(d1);
thresh1 = zeros(n_s,1);
thresh2 = zeros(n_s,1);

% similarity matrix
sim1 = zeros(n_s,n_s);
sim2 = zeros(n_s,n_s);
thresh1(1) = grouping_threshold * min(d1(1,2:n_s));
thresh1(n_s) = grouping_threshold * min(d1(n_s,1:n_s-1));

% loosen up the requirements on distribution........
thresh2(1) = grouping_threshold * min(d2(1,2:n_s));
thresh2(n_s) = grouping_threshold * min(d2(n_s,1:n_s-1));

for ind = 2:n_s-1
    thresh1(ind) = grouping_threshold * min(d1(ind,1:ind-1));
    thresh1(ind) = min(thresh1(ind),grouping_threshold *...
                          min(d1(ind,ind+1:n_s)));
    thresh2(ind) = grouping_threshold * min(d2(ind,1:ind-1));
    thresh2(ind) = min(thresh2(ind), grouping_threshold *...
                          min(d2(ind,ind+1:n_s)));
end

for i = 1:n_s
    for j = 1:n_s
        if d1(i,j) < min(thresh1(i),thresh1(j))
            sim1(i,j) = 1;
            sim1(j,i) = 1;
        end
        if d2(i,j) < min(thresh2(i),thresh2(j))
            sim2(i,j) = 1;
            sim2(j,i) = 1;
        end
    end
end

related = zeros(n_s,n_s);
for i = 1:n_s
    for j = 1:n_s
        % d1 || d2 preferable over d1 && d2
```

```
        if sim1(i,j) == 1 && sim2(i,j) == 1
            related(i,j) = 1;
        end
    end
end

for i = 1:n_s
    sub = find(related(i,:)==1);
    for j = 1:length(sub)
        for k = 1:length(sub)
            related(sub(j),sub(k))=1;
        end
    end
end



function angle_out = posori(angle_in)
%
angle = angle_in;
angle_out = angle;
[rows,cols] = size(angle_in);
for r = 1:rows
   for c = 1:cols
      anglerc = angle(r,c);
      while anglerc>2*pi
          anglerc = anglerc - 2*pi;
      end
      while anglerc < 0
         anglerc = anglerc + 2*pi;
      end
      angle_out(r,c) = anglerc;
   end
end



//Test functions
function  [ratio] = test2a();

num = 10;
interval = 1;
start = 1;
sets = 20;
range = 0;
ratio = zeros(num/interval,sets);

for num_min = start+range:interval:num+range
    for ind = 1:sets
        motors = zeros(10*60*num_min,2);
        [t,~] = size(motors);

        for jnd = 0:t/10-1
            tmp =  -1 + 2*rand(1,2);
            for j = 1:10
                motors(10*jnd+j,:) = tmp;
            end
        end

        x = 0.1; y =0.1; theta = 0;
        s_m = PIE_robot2(x,y,theta,motors,num_min,1,4,6);
```

```
        [d1,d2] = distance_metric(s_m,10);
        % subGroup2: this is the original version of the author without any
        % modification or relaxation
        [related,sim1,sim2] = PIE_subGroup2(d1,d2,2);
        ratio((num_min-range)/interval,ind) = correct_TH(related);
    end
end

fid = fopen('test2a.txt','w');
fid2 = fopen('test2a_10_20.txt','w');
fprintf(fid,'  &');
fprintf(fid2,'  &');

for i = 1:sets/2
    fprintf(fid,'set %d&',i);
    fprintf(fid2,'set %d&',i+10);
end

cases = 10;
fprintf(fid,'\\\\');
fprintf(fid,' \\hline');
fprintf(fid2,'\\\\');
fprintf(fid2,' \\hline');
% set 1 to 10 data
for i = 1:cases
    fprintf(fid,' case %d&',i);
    fprintf(fid2,' case %d&',i+10);
    for j = 1:sets/2-1
        fprintf(fid,'%1.4f&',ratio(i,j));
        fprintf(fid2,'%1.4f&',ratio(i,j+10));
    end
    fprintf(fid,'%1.4f\\\\',ratio(i,sets/2));
    fprintf(fid,'\\hline');
    fprintf(fid2,'%1.4f\\\\',ratio(i,sets));
    fprintf(fid2,'\\hline');
end
fclose(fid);

y = zeros(cases,1);
z = zeros(cases,1);
for x = 1:cases
    y(x) = sum(ratio(x,:))/sets;
    z(x) = var(ratio(x,:));
end
ConfidenceInterval95per = 1.96*z/sqrt(sets);
figure(1); errorbar(y,ConfidenceInterval95per);
xlabel('Time (min)');
ylabel('Grouping Correctness');
axis([0 12 0 1.25]);
print -deps grouping_correctness_10.eps



function  [ratio] = test2b();

num = 10;
interval = 1;
start = 1;
sets = 20;
cases = 10;
range = 0;
```

```
ratio{1} = zeros(num/interval,sets);
ratio{2} = zeros(num/interval,sets);
figure;
for bins = [10 100]
    logInd = log10(bins);
    for num_min = start+range:interval:num+range
        for ind = 1:sets
            motors = zeros(10*60*num_min,2);
            [t,~] = size(motors);

            for jnd = 0:t/10-1
                tmp =  -1 + 2*rand(1,2);
                for j = 1:10
                    motors(10*jnd+j,:) = tmp;
                end
            end

            x = 0.1; y =0.1; theta = 0;
            s_m = PIE_run_robot(x,y,theta,motors,1);
  %          s_m = PIE_run_robot(x,y,theta,motors,num_min,1,4,6);

            [d1,d2] = distance_metric(s_m,bins);

            if bins == 100
             [d1,d2] = distance_metric(s_m,bins/2);
            end
            % subGroup2: this is the original version of the author without any
            % modification or relaxation
            [related,sim1,sim2] = PIE_subGroup2(d1,d2,2);
            ratio{logInd}((num_min-range)/interval,ind) = correct_TH(related);
        end
    end

    if bins == 100
        fid = fopen('test2b.txt','w');
        fid2 = fopen('test2b_10_20.txt','w');
        for i = 1:sets/2
            fprintf(fid,'set %d&',i);
            fprintf(fid2,'set %d&',i+10);
        end

        cases = 10;
        fprintf(fid,'\\\\');
        fprintf(fid,' \\hline');
        fprintf(fid2,'\\\\');
        fprintf(fid2,' \\hline');
        % set 1 to 10 data
        for i = 1:cases
            fprintf(fid,' case %d&',i);
            fprintf(fid2,' case %d&',i+10);
            for j = 1:sets/2-1
                fprintf(fid,'%1.4f&',ratio{logInd}(i,j));
                fprintf(fid2,'%1.4f&',ratio{logInd}(i,j+10));
            end
            fprintf(fid,'%1.4f\\\\',ratio{logInd}(i,sets/2));
            fprintf(fid,'\\hline');
            fprintf(fid2,'%1.4f\\\\',ratio{logInd}(i,sets));
            fprintf(fid2,'\\hline');
        end
        fclose(fid);
    end
```

```
    y = zeros(cases,1);
    z = zeros(cases,1);
    for x = 1:cases
        y(x) = sum(ratio{logInd}(x,:))/sets;
        z(x) = var(ratio{logInd}(x,:));
    end
    ConfidenceInterval95per = 1.96*z/sqrt(sets);
    errorbar(y,ConfidenceInterval95per);
    text(4,y(4),['\leftarrow ' num2str(bins) ' bins'],...
     'HorizontalAlignment','left');
    xlabel('Time (min)');
    ylabel('Grouping Correctness');
    axis([0 12 0 1.25]);
    hold on;
end

    print -deps binComp.eps
```

```
function [d1,d2] = test2c();
%
% pick 100 minutes, and plot d1,d2 for sensors
% 1&2
% 1&13
% 1&25
% 1&26
% 25&26
% 26&27
%

num_min = 10;
motors = zeros(10*60*num_min,2);
[t,~] = size(motors);

for jnd = 0:t/10-1
    for j = 1:10
        motors(10*jnd+j,:) = -1 + 2*rand(1,2);
    end
end

x = 0.1; y =0.1; theta = 0;
s_m = PIE_robot2(x,y,theta,motors,num_min,1,4,6);

[d1,d2] = distance_metric(s_m,10);

% d1 1&2
figure(1);
plot(d1(1,:));
text(12,d1(1,12),'\leftarrow Sensor 1',...
     'HorizontalAlignment','left');
hold on;
plot(d1(2,:),'--');
text(18,d1(2,18),'\leftarrow Sensor 2',...
     'HorizontalAlignment','left');
xlabel('29 Sensors');
ylabel('d2 Metric');
print -deps d1OneAndTwo.eps

% d1 25&26
figure(2);
```

```
plot(d1(25,:));
text(12,d1(25,12),'\leftarrow Sensor 25',...
     'HorizontalAlignment','left');
hold on;
plot(d1(26,:),'--');
text(18,d1(26,18),'\leftarrow Sensor 26',...
     'HorizontalAlignment','left');
xlabel('29 Sensors');
ylabel('d1 Metric');
print -deps d1EastAndNorth.eps

% d2 1&2
figure(3);
plot(d2(1,:));
text(12,d2(1,12),'\leftarrow Sensor 1',...
     'HorizontalAlignment','left');
hold on;
plot(d2(2,:),'--');
text(18,d2(2,18),'\leftarrow Sensor 2',...
     'HorizontalAlignment','left');
xlabel('29 Sensors');
ylabel('d2 Metric');
print -deps d2OneAndTwo.eps

% d2 25&26
figure(4);
plot(d2(25,:));
text(12,d2(25,12),'\leftarrow Sensor 25',...
     'HorizontalAlignment','left');
hold on;
plot(d2(26,:),'--');
text(18,d2(26,18),'\leftarrow Sensor 26',...
     'HorizontalAlignment','left');
xlabel('29 Sensors');
ylabel('d2 Metric');
print -deps d2EastAndNorth.eps



function [ratio,aveR] = test3();
%
% Grouping measure: threshold
% Changes the grouping threshold 1:0.5:4,
% with 10 trials each in 20 minutes of time
%

num_min = 20;
cases = 40;
sets = 10;
ratio = zeros(cases,sets);
aveR = zeros(cases,1);
for k = 1:cases
    grouping_threshold = 0.5*k+0.5;
    for ind = 1:sets
            motors = zeros(10*60*num_min,2);
            [t,~] = size(motors);

            for jnd = 0:t/10-1
                for j = 1:10
                    motors(10*jnd+j,:) = -1 + 2*rand(1,2);
                end
            end
```

```
            x = 1; y =1; theta = 0;
            s_m = PIE_robot2(x,y,theta,motors,num_min,1,4,6);

            [d1,d2] = distance_metric(s_m,10);
            [related,sim1,sim2] = PIE_subGroup2(d1,d2,grouping_threshold);
            %imshow(related);
            ratio(k,ind) = correct_TH(related);
            %disp(['grouping_threshold' num2str(grouping_threshold) '+' num2str(ind)]);
    end
    aveR(k) = sum(ratio(k,:))/10;
end

fid = fopen('test3.txt','w');
fprintf(fid,'  &');

for i = 1:sets
    fprintf(fid,'set %d&',i);
end

fprintf(fid,'\\\\');
fprintf(fid,' \\hline');

for i = 1:cases
    fprintf(fid,' case %d&',i);
    for j = 1:sets-1
        fprintf(fid,'%1.4f&',ratio(i,j));
    end
    fprintf(fid,'%1.4f\\\\',ratio(i,sets));
    fprintf(fid,'\\hline');
end
fclose(fid);

y = zeros(cases,1);
z = zeros(cases,1);
for x = 1:cases
    y(x) = sum(ratio(x,:))/sets;
    z(x) = var(ratio(x,:));
end
kk = 1:cases;
grouping_threshold = 0.5*kk+0.5;
ConfidenceInterval95per = 1.96*z/sqrt(sets);
figure(1); errorbar(grouping_threshold,y,ConfidenceInterval95per);
xlabel('Similarity Measure');
ylabel('Grouping Correctness');
axis([1 20.5 0 1.25]);
print -deps SimilarityThreshold.eps




function ratio2 = test4();
%
%   Grouping measure: environment
%   Make the width and height of the room as parameters.
%   Draw the trails of the image. See if the pixels have been visited.
%

sets = 10;
cases = 3;
for x_max = [3 4 6]
```

```
    for k = 1:sets
    if x_max == 3
        y_max = 4;i=1;
    elseif x_max == 4
        y_max = 6;i=2;
    else
        y_max = 8;i=3;
    end

    num_min = 20;
    motors = zeros(10*60*num_min,2);
    [t,~] = size(motors);

    for jnd = 0:t/10-1
        for j = 1:10
            motors(10*jnd+j,:) = -1 + 2*rand(1,2);
        end
    end
    x = 1; y =1; theta = 0;
    s_m{i} = PIE_robot2(x,y,theta,motors,num_min,1,x_max,y_max);
    [d1,d2] = distance_metric(s_m{i},10);
    [related,sim1,sim2] = PIE_subGroup2(d1,d2,2);
    ratio2(i,k) = correct_TH(related);
    end
end

fid = fopen('test4.txt','w');
fprintf(fid,'  &');

for i = 1:sets
    fprintf(fid,'set %d&',i);
end

fprintf(fid,'\\\\');
fprintf(fid,' \\hline');

for i = 1:cases
    fprintf(fid,' case %d&',i);
    for j = 1:sets-1
        fprintf(fid,'%1.4f&',ratio2(i,j));
    end
    fprintf(fid,'%1.4f\\\\',ratio2(i,sets));
    fprintf(fid,'\\hline');
end
fclose(fid);

for i = 1:cases
    ave(i) = sum(ratio2(i,:))/10;
end

y = zeros(cases,1);
z = zeros(cases,1);
for x = 1:cases
    y(x) = sum(ratio2(x,:))/sets;
    z(x) = var(ratio2(x,:));
end

ConfidenceInterval95per = 1.96*z/sqrt(sets);
tmp = [3*4 4*6 6*8];
figure; errorbar(tmp,y,ConfidenceInterval95per);
axis([0 50 0 1.25]);
xlabel('Area (Square Meters)');
```

```
ylabel('Grouping Correctness');

print -deps EnvironmentSize.eps




function aveR = test5_main();
%
% Grouping measure varied by sensor noise
% generate 20 minutes trajectory with noises
% a. uniform noise: Vu = V + U(-beta,beta)
%             beta = [0.2 0.4 0.6 0.8 1]
% b. Gaussian noise: Vg = V + N(0,sigma^2)
%             sigma^2 = [0.001 0.01 0.1 0.5 1]
%

num_min = 10;
cases = 6;
sets = 10;

ratio = zeros(cases,sets);
aveR = zeros(cases,1);

for ind = 1:sets
    motors = zeros(10*60*num_min,2);
    [t,~] = size(motors);

    for jnd = 0:t/10-1
        for j = 1:10
            motors(10*jnd+j,:) = -1 + 2*rand(1,2);
        end
    end

    x = 1; y =1; theta = 0;
    s_m = PIE_robot2(x,y,theta,motors,num_min,1,4,6);
    [m n] = size(s_m);
    for k = 1:cases
        sm = s_m + 0.2*(k-1)*(-1+2*rand(m,n));
        [d1,d2] = distance_metric(sm,10);
        related = PIE_subGroup2(d1,d2,2);
        %imshow(related);
        ratio(k,ind) = correct_TH(related);
    end
end

for k = 1:cases
    aveR(k) = sum(ratio(k,:))/10;
end

fid = fopen('test5a.txt','w');
fprintf(fid,'  &');

for i = 1:sets
    fprintf(fid,'set %d&',i);
end

fprintf(fid,'\\\\');
fprintf(fid,' \\hline');

for i = 1:cases
```

```
    fprintf(fid,' case %d&', i);
    for j = 1:sets-1
        fprintf(fid,'%1.4f&',ratio(i,j));
    end
    fprintf(fid,'%1.4f\\\\',ratio(i,sets));
    fprintf(fid,'\\hline');
end
fclose(fid);

y = zeros(cases,1);
z = zeros(cases,1);
for x = 1:cases
    y(x) = sum(ratio(x,:))/sets;
    z(x) = var(ratio(x,:));
end

ConfidenceInterval95per = 1.96*z/sqrt(sets);
xx = [0 0.2 0.4 0.6 0.8 1];
figure(1); errorbar(xx,y,ConfidenceInterval95per);
xlabel('Noise Magnitude');
ylabel('Grouping Correctness');
axis([-0.1 1.1 0 1.25]);
print -deps UniformNoise.eps
```

```
function aveR = test5b_main();
%
% Grouping measure varied by sensor noise
% a. uniform noise: Vu = V + U(-beta,beta)
%      beta = [0.2 0.4 0.6 0.8 1]
% b. Gaussian noise: Vg = V + N(0,sigma^2)
%      sigma^2 = [0.001 0.01 0.1 0.5 1]
%

num_min = 10;
cases = 6;
sets = 10;
ratio = zeros(cases,sets);

for ind = 1:sets
    motors = zeros(10*60*num_min,2);
    [t,~] = size(motors);

    for jnd = 0:t/10-1
        for j = 1:10
            motors(10*jnd+j,:) = -1 + 2*rand(1,2);
        end
    end

    x = 1; y =1; theta = 0;
    s_m = PIE_robot2(x,y,theta,motors,num_min,1,4,6);
    [m n] = size(s_m);
    for k = 1:cases
        sm = s_m;
        for i = 1:m
            for jnd = 1:n
                x = sm(i,jnd);
                if k==2
                    sm(i,jnd) =  x + 1/sqrt(2*pi*0.001)*exp(-1/2*x^2/0.001);
```

```
                elseif k==3
                    sm(i,jnd) =  x + 1/sqrt(2*pi*0.01)*exp(-1/2*x^2/0.01);
                elseif k==4
                    sm(i,jnd) =  x + 1/sqrt(2*pi*0.1)*exp(-1/2*x^2/0.1);
                elseif k==5
                    sm(i,jnd) =  x + 1/sqrt(2*pi*0.5)*exp(-1/2*x^2/0.5);
                else
                    sm(i,jnd) =  x + 1/sqrt(2*pi*1)*exp(-1/2*x^2/1);
                end
            end
        end
        [d1,d2] = distance_metric(sm,10);
        related = PIE_subGroup2(d1,d2,2);
        %imshow(related);
        ratio(k,ind) = correct_TH(related);
    end
end

fid = fopen('test5b.txt','w');
fprintf(fid,'  &');

for i = 1:sets
    fprintf(fid,'set %d&',i);
end

fprintf(fid,'\\\\');
fprintf(fid,' \\hline');

for i = 1:cases
    fprintf(fid,' case %d&',i);
    for j = 1:sets-1
        fprintf(fid,'%1.4f&',ratio(i,j));
    end
    fprintf(fid,'%1.4f\\\\',ratio(i,sets));
    fprintf(fid,'\\hline');
end
fclose(fid);

y = zeros(cases,1);
z = zeros(cases,1);
for x = 1:cases
    y(x) = sum(ratio(x,:))/sets;
    z(x) = var(ratio(x,:));
end

ConfidenceInterval95per = 1.96*z/sqrt(sets);
xx = [0 0.001 0.01 0.1 0.5 1];
% figure; errorbar(xx,y,ConfidenceInterval95per);
figure; errorbar(y,ConfidenceInterval95per);
xlabel('Noise Magnitude');
ylabel('Grouping Correctness');
% axis([-0.1 1.1 0 1.25]);
axis([0 7 0 1.25]);
print -deps GaussianNoise.eps

figure(2); errorbar(xx,y,ConfidenceInterval95per);
xlabel('Noise Magnitude');
ylabel('Grouping Correctness');
axis([-0.1 1.1 0 1.25]);
print -deps GaussianNoise_2.eps
```

```
function test6();
%
% sensor placement
% put 12 sensors at front and 12 at back with 1 degree angle
%

sets = 10;
cases = 20;
ratio = zeros(cases/2,sets);

for num_min = 2:2:cases
    for ind = 1:sets
        motors = zeros(10*60*num_min,2);
        [t,˜] = size(motors);

        for jnd = 0:t/10-1
            tmp =  -1 + 2*rand(1,2);
            for j = 1:10
                motors(10*jnd+j,:) = tmp;
            end
        end

        x = 1; y =1; theta = 0;
        s_m = PIE_robot2_test6(x,y,theta,motors,num_min,1,4,6);
        [d1,d2] = distance_metric(s_m,10);
        [related] = PIE_subGroup2(d1,d2,2);
        %imshow(related);
        ratio(num_min/2,ind) = correct_TH(related);
    end
end

fid = fopen('test6.txt','w');
fprintf(fid,'  &');

for i = 1:sets
    fprintf(fid,'set %d&',i);
end

fprintf(fid,'\\\\');
fprintf(fid,' \\hline');
for i = 1:cases/2
    fprintf(fid,' case %d&', i);
    for j = 1:sets-1
        fprintf(fid,'%1.4f&',ratio(i,j));
    end
    fprintf(fid,'%1.4f\\\\',ratio(i,sets));
    fprintf(fid,'\\hline');
end
fclose(fid);

y = zeros(cases/2,1);
z = zeros(cases/2,1);
for x = 1:cases/2
    y(x) = sum(ratio(x,:))/sets;
    z(x) = var(ratio(x,:));
end

xx = 2:2:cases;
```

```
ConfidenceInterval95per = 1.96*z/sqrt(sets);
figure; errorbar(y,ConfidenceInterval95per);

xlabel('Time (minutes)');
ylabel('Grouping Correctness');
axis([-1 cases/2+1 0 1.25]);
print -deps sensorPlacement.eps
```

```
function ratio2 = test7();
%
% Grouping Measure maximum range
% Allow the maximum range to change 1:7.
% Do 20 minutes, 10 trials each.
%

num_min = 20;
cases = 9;
sets = 10;
ratio2 = zeros(cases,sets);

for k = 1:cases
    for i = 1:sets
    motors = zeros(10*60*num_min,2);
    [t,~] = size(motors);

    for jnd = 0:t/10-1
        for j = 1:10
            motors(10*jnd+j,:) = -1 + 2*rand(1,2);
        end
    end

    x = 0.1; y =0.1; theta = 0;
    s_m = PIE_robot2(x,y,theta,motors,num_min,k,4,6);
    [d1,d2] = distance_metric(s_m,10);
    [related] = PIE_subGroup2(d1,d2,2);
    ratio2(k,i) = correct_TH(related);
    %disp([num2str(k) '+' num2str(i)]);
    end
end

fid = fopen('test7.txt','w');
fprintf(fid,'  &');

for i = 1:sets
    fprintf(fid,'set %d&',i);
end

fprintf(fid,'\\\\');
fprintf(fid,' \\hline');

for i = 1:cases
    fprintf(fid,' case %d&', i);
    for j = 1:sets-1
        fprintf(fid,'%1.4f&',ratio2(i,j));
    end
    fprintf(fid,'%1.4f\\\\',ratio2(i,sets));
    fprintf(fid,' \\hline');
end
```

```
fclose(fid);

max_range= 1:cases;
y = zeros(cases,1);
z = zeros(cases,1);
for x = 1:cases
    y(x) = sum(ratio2(x,:))/sets;
    z(x) = var(ratio2(x,:));
end
ConfidenceInterval95per = 1.96*z/sqrt(sets);
figure(3); errorbar(max_range,y,ConfidenceInterval95per);
xlabel('Maximum Range (meters)');ylabel('Grouping Correctness');
axis([0 cases+1 0 1.35]);
print -deps sensorRange.eps
```

Appendix 2: Grouping Results

Grouping Result vs Time (10 bins)

as shown in Figure 3 and Figure 4 (10 bin case)

	set 1	set 2	set 3	set 4	set 5	set 6	set 7	set 8	set 9	set 10
case 1	0.8121	0.9976	0.8098	0.8977	0.7384	0.8193	0.6124	0.6908	0.7788	0.4839
case 2	0.9976	1.0000	0.7979	0.9976	0.9976	0.6124	0.5600	0.6623	0.6980	0.5315
case 3	0.6718	1.0000	0.9952	0.8930	0.9976	0.7574	0.9001	0.5981	0.9952	0.9929
case 4	0.4863	0.8121	0.7574	1.0000	0.6504	0.5291	0.6385	0.6885	0.5339	0.5291
case 5	0.5434	0.6694	0.9976	0.9952	0.9952	0.5933	0.6837	0.9976	0.5434	0.9857
case 6	0.9976	0.6932	0.9952	0.5244	0.8454	0.6813	0.9952	0.7313	0.9976	0.6813
case 7	0.6076	0.8502	0.9857	0.9952	0.7289	0.7669	0.9952	0.6338	0.9929	0.9453
case 8	0.7289	0.8549	0.5743	0.9929	0.5553	0.9952	0.6195	0.9952	0.8026	0.7027
case 9	0.6576	0.6314	0.6504	0.6813	0.5505	0.9929	0.7051	0.6409	0.9952	0.9929
case 10	0.8526	0.6029	0.9976	0.8050	0.9976	0.8526	0.6790	0.8526	0.9976	0.9976

	set 11	set 12	set 13	set 14	set 15	set 16	set 17	set 18	set 19	set 20
case 1	0.5767	0.7122	0.7194	0.6361	0.6171	0.7194	1.0000	0.9976	0.5648	0.7693
case 2	0.7455	1.0000	1.0000	0.5838	0.6861	0.5862	0.7551	0.5315	0.6647	0.8549
case 3	0.7337	0.7503	0.9976	0.8169	0.6647	0.7574	0.6885	0.9952	0.5505	1.0000
case 4	0.6124	0.7717	0.6480	0.7313	0.6813	0.9976	0.5743	0.6409	0.6314	0.6314
case 5	0.6314	0.8954	0.6528	0.9929	0.7313	0.7337	0.9857	0.9952	0.9976	0.9952
case 6	0.9976	0.9001	0.9857	0.9929	0.5767	0.9857	0.8074	0.9952	0.9976	0.9952
case 7	0.9976	0.9976	0.6980	0.5600	0.9952	0.9976	0.8478	0.9952	0.9929	0.6409
case 8	0.9952	0.6980	0.9952	0.9976	0.7241	0.9952	0.6932	0.7717	0.9976	0.6338
case 9	0.6718	0.7669	0.9976	0.6290	0.5719	0.9334	0.5981	0.7646	0.9976	0.5529
case 10	0.9929	0.5529	0.9929	0.9952	0.9976	0.9952	0.9976	0.9952	0.9952	0.7812

Grouping Result vs Time (100 bins)

as shown in Figure 4, the 100 bins case

	set 1	set 2	set 3	set 4	set 5	set 6	set 7	set 8	set 9	set 10
case 1	0.7122	0.9976	0.9952	0.6361	0.9929	0.9976	0.9976	0.9952	0.6813	0.9905
case 2	0.9976	0.9976	0.9952	0.9952	0.9976	0.9976	0.9976	0.9976	0.9976	0.9976
case 3	0.9976	0.9952	0.9905	0.9952	0.9976	0.9952	0.9905	0.9905	0.9952	0.9952
case 4	0.9976	0.9834	0.9976	0.9952	0.9952	0.9905	0.9834	0.9952	0.9976	0.9952
case 5	0.9929	0.9952	0.9905	0.9952	0.9834	0.7812	0.9905	0.9952	0.8549	0.9976
case 6	0.9905	0.9952	0.9929	0.9952	0.9952	0.9952	0.9929	0.9905	0.9952	0.9976
case 7	0.9952	0.9929	0.9952	0.9905	0.9929	0.9929	0.9905	0.9929	0.9929	0.9834
case 8	0.9905	0.9929	0.9929	0.9952	0.9929	0.9952	0.9929	0.9952	0.9952	0.9929
case 9	0.9952	0.9905	0.9952	0.9952	0.9952	0.9952	0.9976	0.9976	0.9929	0.9834
case 10	0.9952	0.9952	0.9905	0.9952	0.9929	0.9905	0.9952	0.9929	0.9834	0.9952

	set 11	set 12	set 13	set 14	set 15	set 16	set 17	set 18	set 19	set 20
case 1	0.9952	0.9952	0.9952	0.9976	0.9976	0.6314	0.9952	0.9952	0.8169	0.8145
case 2	0.9952	0.9952	0.9929	0.9976	0.9976	0.9952	0.9976	0.9952	0.9976	0.9976
case 3	0.9834	0.9905	0.9976	0.9952	0.9976	0.9952	0.9976	0.9905	0.9976	0.9905
case 4	0.9905	0.9905	0.9952	0.9952	0.9905	0.9834	0.9929	0.9976	0.9929	0.9952
case 5	0.6790	0.9834	0.9929	0.9952	0.9905	0.9929	0.9952	0.9952	0.9929	0.9834
case 6	0.9929	0.9929	0.9952	0.9976	0.9929	0.9952	0.9952	0.9834	0.9834	0.9929
case 7	0.9929	0.9952	0.9952	0.9952	0.9834	0.9834	0.9952	0.9952	0.9834	0.9834
case 8	0.9929	0.9929	0.9929	0.9905	0.9952	0.9952	0.9952	0.9834	0.9952	0.9905
case 9	0.9952	0.9929	0.9929	0.9952	0.9952	0.9952	0.9952	0.9952	0.9952	0.9929
case 10	0.9952	0.9952	0.9834	0.9952	0.9905	0.9929	0.9952	0.9905	0.9929	0.9834

Grouping Measure: threshold

as shown in Figure 9

	set 1	set 2	set 3	set 4	set 5	set 6	set 7	set 8	set 9	set 10
case 1	0.3983	0.3983	0.3983	0.3983	0.3983	0.3983	0.3983	0.3983	0.3983	0.3983
case 2	0.4197	0.5553	0.4625	0.4816	0.4578	0.4697	0.4340	0.4483	0.4269	0.4174
case 3	0.8930	0.4982	0.5838	0.4721	0.6171	0.4483	0.5315	0.8549	0.8169	0.6457
case 4	0.9976	0.9929	0.9952	0.9857	0.8098	0.9952	0.9929	0.9929	0.6314	0.9952
case 5	0.6694	0.9857	0.9952	0.9857	0.9857	0.9857	0.9857	0.9857	0.6718	0.9952
case 6	0.9952	0.9952	0.8859	0.9857	0.9952	0.9857	0.9952	0.9929	0.9929	0.9952
case 7	0.9857	0.9952	0.9929	0.9952	0.9952	0.9857	0.9857	0.7015	0.9857	0.9952
case 8	0.9834	0.9952	0.9952	0.9952	0.9952	0.9834	0.9952	0.9857	0.9857	0.9976
case 9	0.9952	0.9857	0.5933	0.9857	0.9857	0.9857	0.9952	0.9857	0.9762	0.9952
case 10	0.9952	0.9952	0.9857	0.9952	0.9952	0.9762	0.6385	0.9857	0.4625	0.9857
case 11	0.9762	0.9952	0.9834	0.9952	0.9952	0.9857	0.9857	0.9857	0.9857	0.9857
case 12	0.9762	0.9952	0.9929	0.9857	0.8835	0.9857	0.9952	0.9857	0.9857	0.9834
case 13	0.9857	0.9857	0.9857	0.9857	0.9857	0.9857	0.9762	0.9857	0.9857	0.9952
case 14	0.9857	0.9857	0.9834	0.9929	0.4530	0.9952	0.9952	0.9857	0.9762	0.9952
case 15	0.9358	0.9405	0.9857	0.9857	0.9952	0.9857	0.9857	0.9857	0.9857	0.9762
case 16	0.8835	0.6361	0.9857	0.9857	0.8264	0.9857	0.9857	0.9762	0.9834	0.9215
case 17	0.9857	0.9310	0.9857	0.9905	0.9857	0.8799	0.9952	0.9857	0.6231	0.8740
case 18	0.9857	0.9834	0.9762	0.9834	0.9857	0.9952	0.9857	0.9762	0.9215	0.7646
case 19	0.8835	0.9857	0.9905	0.9857	0.9762	0.6361	0.9952	0.9857	0.8216	0.9834
case 20	0.9643	0.9952	0.9857	0.9857	0.9905	0.9857	0.9857	0.9857	0.9857	0.9905
case 21	0.9310	0.9215	0.9857	0.9215	0.9762	0.9952	0.9762	0.9905	0.9857	0.9310
case 22	0.9762	0.9762	0.9857	0.9762	0.9762	0.9857	0.8740	0.9310	0.6361	0.6361
case 23	0.8740	0.8740	0.9857	0.9762	0.9762	0.8740	0.9857	0.6361	0.9215	0.9405
case 24	0.9952	0.9857	0.7027	0.9762	0.9762	0.9215	0.9762	0.8312	0.9762	0.9857
case 25	0.9215	0.9762	0.6361	0.9762	0.9905	0.9762	0.9857	0.8740	0.9762	0.6361
case 26	0.6361	0.9762	0.9762	0.9405	0.9762	0.7646	0.9215	0.9952	0.9762	0.9762
case 27	0.9310	0.9762	0.9952	0.6361	0.8740	0.9215	0.8740	0.6361	0.9762	0.9952
case 28	0.9857	0.9905	0.6361	0.6361	0.9952	0.7646	0.9857	0.9358	0.9857	0.9762
case 29	0.9762	0.6992	0.9857	0.9857	0.9643	0.9405	0.9762	0.9857	0.9857	0.9762
case 30	0.6361	0.9762	0.8740	0.6361	0.8740	0.6361	0.8835	0.7646	0.7848	0.9762
case 31	0.6361	0.5898	0.9762	0.6361	0.9215	0.6361	0.7408	0.6361	0.9762	0.8740
case 32	0.8740	0.6361	0.9952	0.9762	0.7646	0.9857	0.9762	0.6361	0.9952	0.9405
case 33	0.6361	0.9762	0.9857	0.9762	0.4328	0.9905	0.6361	0.6361	0.9762	0.6361
case 34	0.9310	0.6361	0.9358	0.6361	0.9762	0.9762	0.9857	0.9762	0.9952	0.9762
case 35	0.9857	0.9762	0.9762	0.9215	0.5268	0.6361	0.9857	0.9762	0.8216	0.9762
case 36	0.6361	0.9857	0.8740	0.9643	0.9857	0.9215	0.6361	0.9952	0.9952	0.6361
case 37	0.9762	0.7408	0.5565	0.9310	0.9762	0.9857	0.6361	0.9762	0.9762	0.9762
case 38	0.9905	0.9762	0.7848	0.9905	0.6361	0.6361	0.6361	0.6361	0.9762	0.6361
case 39	0.9762	0.9405	0.6361	0.6361	0.9215	0.6361	0.9857	0.9762	0.9762	0.6361
case 40	0.9762	0.6361	0.6361	0.9215	0.4031	0.6361	0.7420	0.9762	0.6361	0.8799

Grouping Measure: Environment

as shown in Figure 10

	set 1	set 2	set 3	set 4	set 5	set 6	set 7	set 8	set 9	set 10
case 1	0.8502	0.9857	0.5434	0.6956	0.7289	0.7004	0.6885	0.7004	0.5696	0.6385
case 2	0.4935	0.3817	0.5339	0.8502	0.5731	0.4756	0.6195	0.5220	0.4697	0.4102
case 3	0.6290	0.7075	0.4554	0.9952	0.9929	0.9952	0.6861	0.5981	0.6076	0.3995

Grouping Measure: Uniform Noise

as shown in Figure 11

	set 1	set 2	set 3	set 4	set 5	set 6	set 7	set 8	set 9	set 10
case 1	0.4483	0.6266	0.6124	0.6076	0.4721	0.7408	0.4233	0.4340	0.6837	0.4483
case 2	0.6528	0.5957	0.8859	0.8859	0.5196	0.8954	0.5220	0.5767	0.8977	0.7146
case 3	0.8050	0.9857	0.9857	0.9857	0.9405	0.8954	0.5220	0.6623	0.9905	0.6694
case 4	0.9310	0.6361	0.9857	0.9857	0.6528	0.8526	0.7574	0.9405	0.9952	0.9429
case 5	0.9929	0.9834	0.8835	0.9834	0.8835	0.9382	0.8811	0.8835	0.9857	0.8835
case 6	0.8835	0.9834	0.9905	0.9834	0.8835	0.9834	0.8811	0.7122	0.7574	0.8835

Grouping Measure: Gaussian Noise

as shown in Figure 12

	set 1	set 2	set 3	set 4	set 5	set 6	set 7	set 8	set 9	set 10
case 1	0.6956	0.6314	0.6076	0.4911	0.5600	0.7836	0.6980	0.5315	0.5600	0.4958
case 2	0.6623	0.6314	0.6076	0.4911	0.6885	0.5838	0.6980	0.5315	0.5577	0.4958
case 3	0.6623	0.6314	0.6076	0.4911	0.9976	0.5220	0.6980	0.5315	0.5577	0.4958
case 4	0.7146	0.6861	0.9976	0.4625	0.5505	0.9976	0.8169	0.4649	0.6147	0.6076
case 5	0.6171	0.5672	0.6076	0.4697	0.6885	0.9976	0.5791	0.5315	0.5577	0.5339
case 6	0.6956	0.6314	0.6076	0.4911	0.5600	0.7836	0.6980	0.5315	0.5600	0.4958

Grouping Measure: Sensor Placement

as shown in Figure 13

	set 1	set 2	set 3	set 4	set 5	set 6	set 7	set 8	set 9	set 10
case 1	0.5767	0.5886	0.3674	0.5815	0.5624	0.6219	0.6243	0.4721	0.4221	0.5220
case 2	0.5196	0.5862	0.5862	0.5006	0.5339	0.3650	0.4982	0.6790	0.5600	0.4887
case 3	0.5624	0.4839	0.4744	0.4530	0.5315	0.6100	0.6243	0.6243	0.5220	0.5624
case 4	0.5101	0.4816	0.5577	0.6243	0.6266	0.5196	0.6243	0.5815	0.5196	0.5054
case 5	0.5268	0.5482	0.5624	0.5291	0.5386	0.6266	0.5553	0.5054	0.6100	0.6361
case 6	0.5696	0.4768	0.5268	0.5767	0.5149	0.5767	0.5101	0.4863	0.4863	0.4863
case 7	0.5862	0.4958	0.5339	0.4911	0.5482	0.5981	0.5125	0.6100	0.5910	0.4911
case 8	0.5149	0.5815	0.5101	0.5577	0.5125	0.5577	0.6790	0.4816	0.5386	0.5101
case 9	0.6409	0.5505	0.5910	0.5600	0.5410	0.5434	0.5505	0.5672	0.5981	0.5220
case 10	0.6219	0.5101	0.5767	0.5101	0.5981	0.5624	0.5268	0.5886	0.5410	0.5339

Grouping Measure: Maximum Range

as shown in Figure 14

	set 1	set 2	set 3	set 4	set 5	set 6	set 7	set 8	set 9	set 10
case 1	0.6504	0.9976	0.6504	0.9976	0.5458	0.6147	0.9976	0.6908	0.5482	0.7313
case 2	0.9952	0.9929	0.9857	0.9976	0.5125	0.6243	0.6385	0.6908	0.5672	0.8954
case 3	0.9857	0.4887	0.5482	0.6623	0.6076	0.6766	0.5886	0.6813	0.9929	0.9952
case 4	0.9952	0.6980	0.5315	0.5600	0.6790	0.5482	0.5791	0.6861	0.5244	0.6813
case 5	0.8145	0.9952	0.9952	0.8859	0.9952	0.7669	0.4935	0.9952	0.5910	0.6409
case 6	0.6338	0.7812	0.9952	0.6314	0.5791	0.8930	0.6290	0.7004	0.6861	0.9952
case 7	0.6766	0.7836	0.9976	0.5363	0.9952	0.9976	0.9952	0.9952	0.7527	0.7122
case 8	0.8954	0.9976	0.9857	0.8526	0.9952	0.9857	0.6433	0.6766	0.9857	0.7241
case 9	0.7551	0.9952	0.9952	0.9952	0.4911	0.9952	0.9929	0.6790	0.6409	0.6480

References

[1] David M. Pierce. *Map Learning with Uninterpreted Sensors and Effectors*. PhD thesis, University of Texas, Austin, Austin, Texas, May 1995.