MTRX5700

Experimental Robotics

Assignment 2

Author(s):
Kausthub Krishnamurthy
James Ferris
Sachith Gunawardhana

SID:
312086040
311220045
440623630

Due: March 25, 2015

1 Question 1

1.a

1.b

1.c

Code Listing

See Appendix A [9.1]

2 Question 2

2.a Validity Check

2.a.i R_1

2.a.ii R_2

2.a.iii R_3

2.a.iv R_4

2.b Roll/Pitch/Yaw Angles

2.c Angle Estimation

Code Listing

See Appendix A [9.2]

3 Question 3

The Iterated Closed Loop Algorithm

By modifying the given code pieces, we will implement an Iterated Closed Loop algorithm to estimate the position of a vehicle as it moves through its surroundings.
All code pieces, original and modified, can be found in Appendix A [3].

3..i Part A - Implementing the ICP

By modifying the given showICP.m file, we will exam the resultant ICP features generated for a single data set. The set in question is frame 500, and we will use frame 520 as our initial 'guess'.
Firstly, using the default variables of a grid size of 0.005, and a maximum iterative loop of 40, we can generate the following graph:

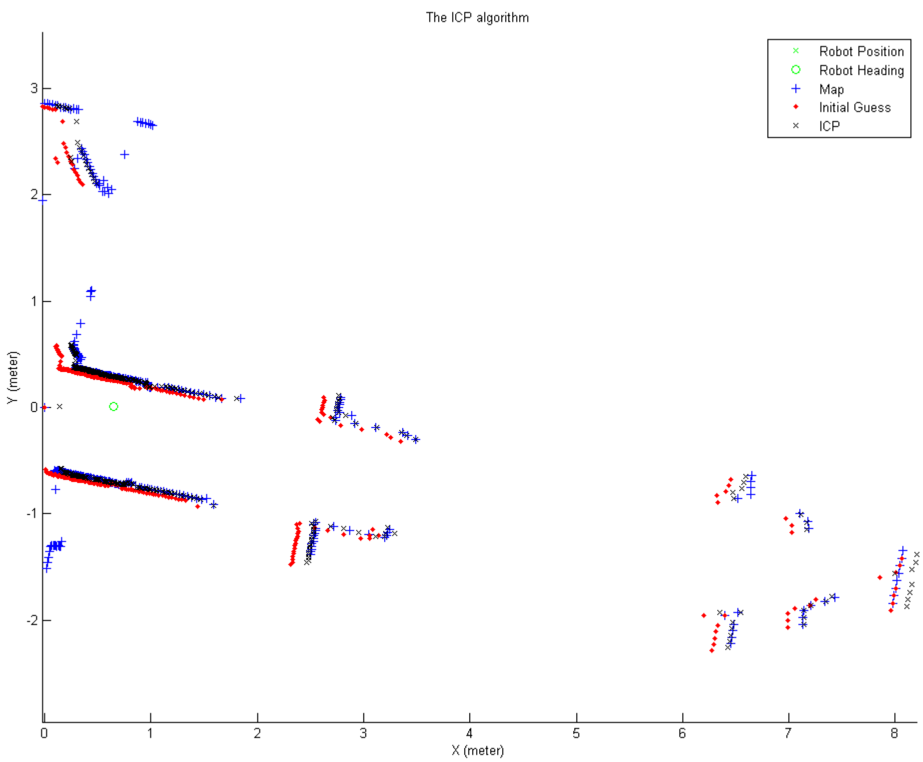


Figure 1: ICP estimate for maximum iterations of 40 and grid size of 0.005

We will now examine the effect of modifying some of the variables of the ICPv4.m algorithm.
Firstly we will look at changing the grid size. For a smaller grid size of 0.001 we obtain the following graph:

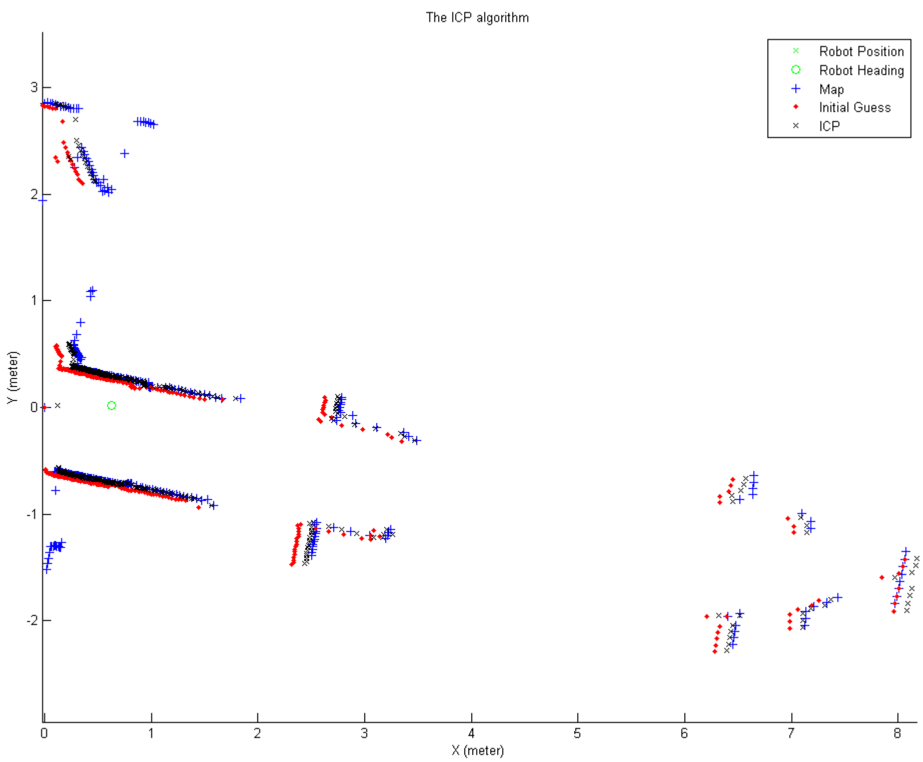


Figure 2: ICP estimate for maximum iterations of 40 and grid size of 0.001

Next, for a grid size of 0.1:

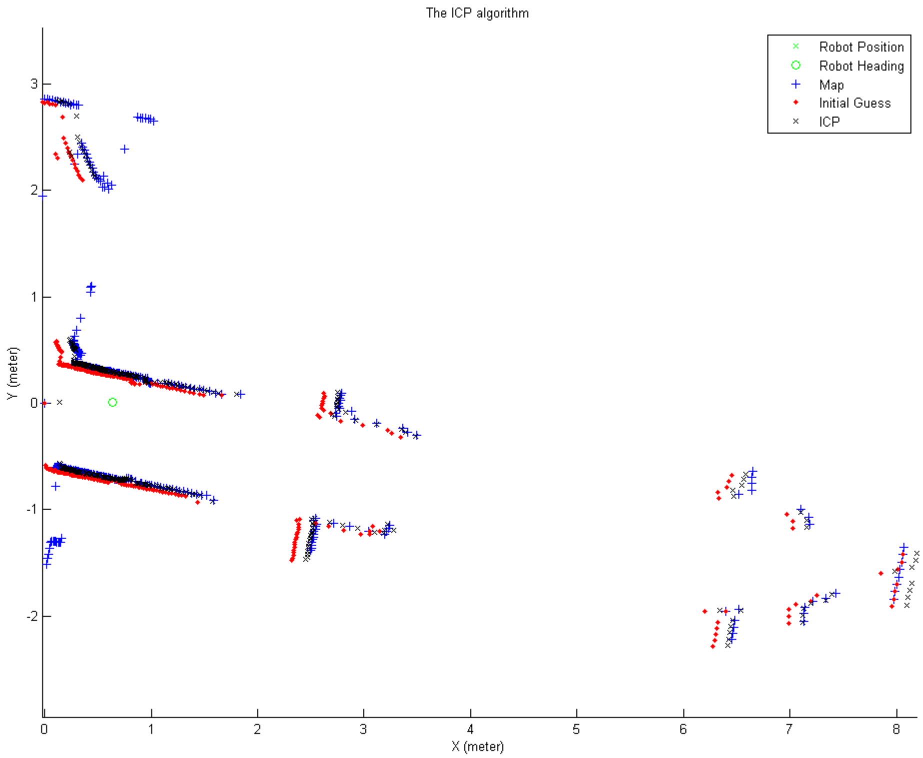


Figure 3: ICP estimate for maximum iterations of 40 and grid size of 0.005

As can be seen, changing the grid size has little effect on the overall ICP .
It does, however, have an affect on the number of collision points detected. For a grid size of 0.005, 188 points collide. For 0.001, 32 points collide, and for 0.01, 252 points collide. This is expected - an increase in in the grid size means a large sample section, with a greater likelihood of multiple points landing in a grid.

Checking for matching pairs reveals an interesting point - for all tested values for grid size, the number of matching points is the same - 362. Also of worthy note, despite a maximum number of iterations of 40, no more than 9 iterations are used. Changing the maximum number of iterations to 10 resulted in no changes to any of the previous tests. As such, the maximum iterative size does not need to be nearly so large.

Looking at the generated deltaPose$_bar, we can get an idea of the estimated heading of the vehicle, and by looking at deltaPose_bar_cov$
$The pose was as follows:$

$deltaPose_{bar} = [0.1360, 0.0116, 0.0004]$ where the pattern is $[x, y, \theta]$
This Is very close to the zero position, which is to be expected seeing as this ICP algorithm has only taken a single frame. Relative movement should be little at this point.
The Pose covariance for this is as follows:

$$deltaPose_bar_cov = 1.0^-5 * \begin{pmatrix} 0.2924 & 0.0130 & -0.0099 \\ 0.0130 & 0.4039 & -0.0863 \\ -0.0099 & -0.0863 & 0.0659 \end{pmatrix}$$

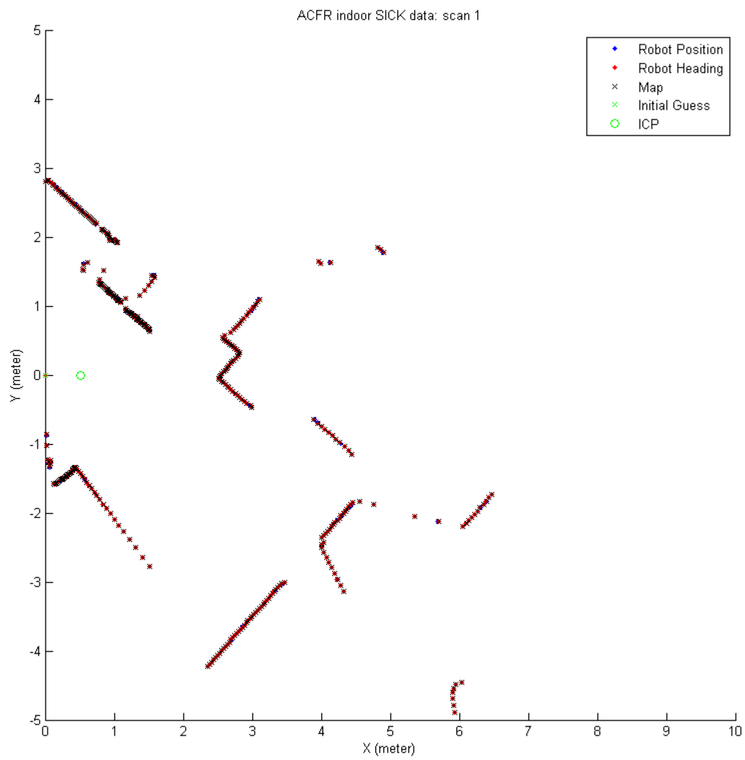
As can be seen, the covariance matrix is extremely close to zero. This is indicative of the factors involved being completely independent, though it does not confirm this. Again, seeing as this is run from a single frame, it is not an indicator for the overall relationship - we have used far too few data points to be able to rule anything out.

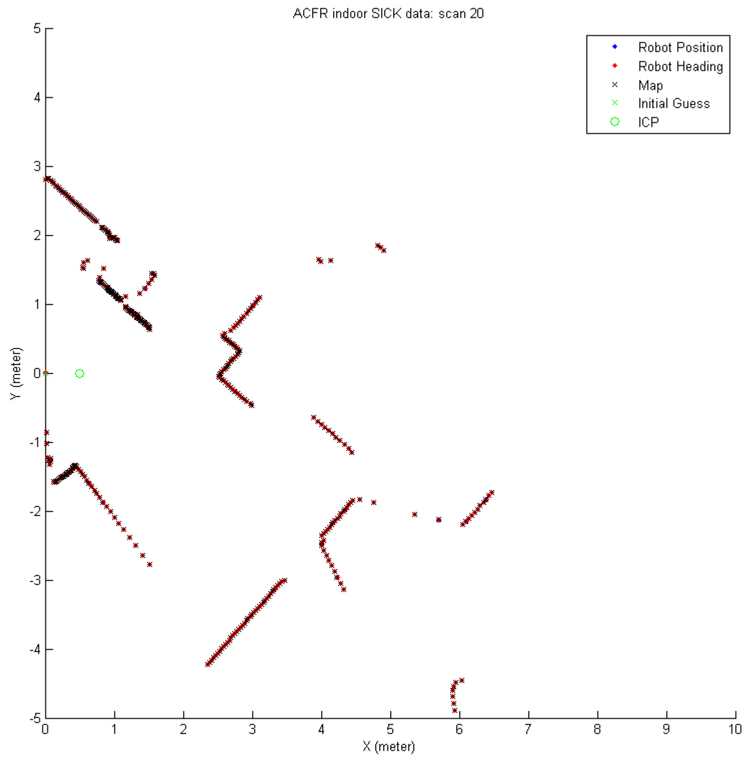
Problems with the Algorithm

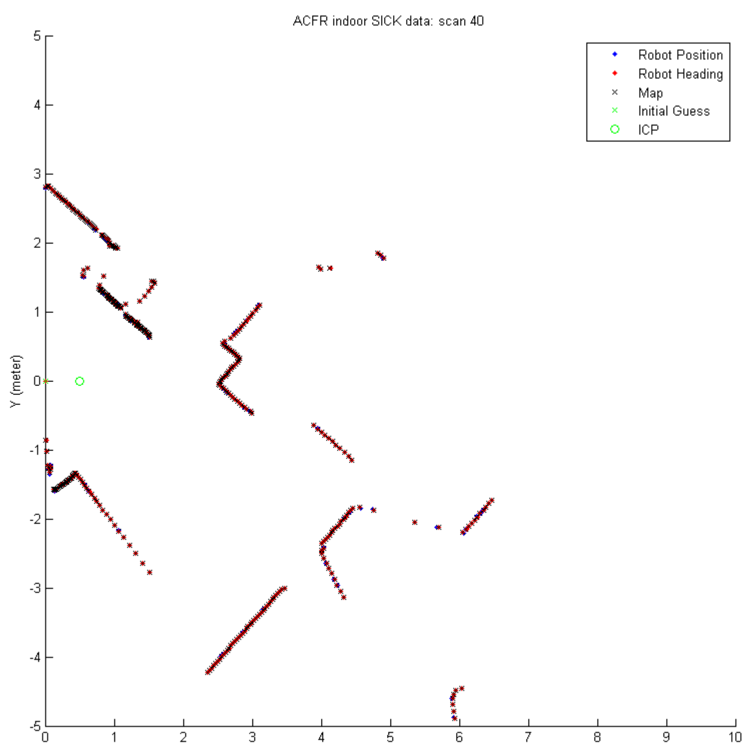
ICPv4.m takes little into account to do with angles/rotations, mostly using x and y values. This would mean that any rotational movement in the map would be taken poorly into account, as will be evident further on.

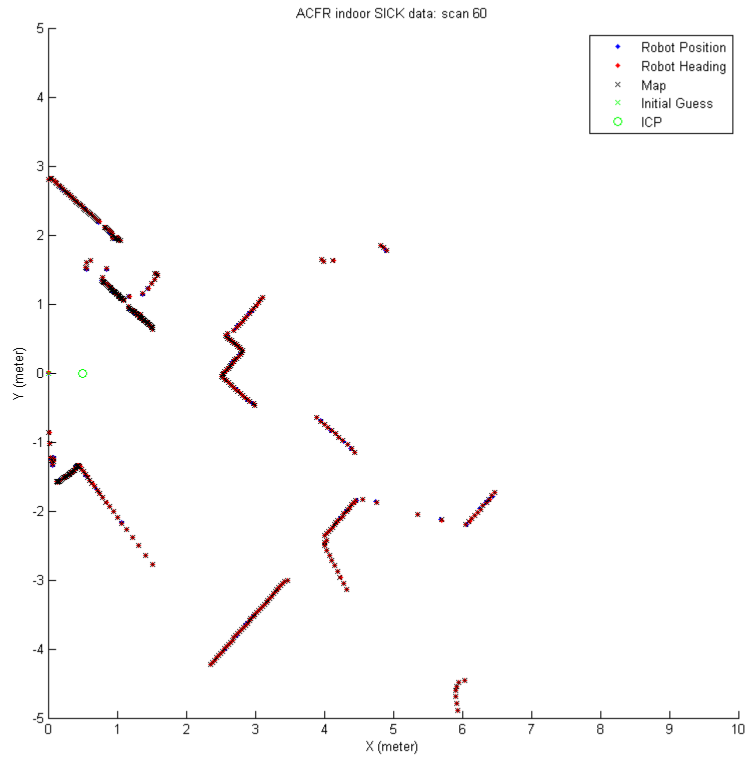
3..ii PartB

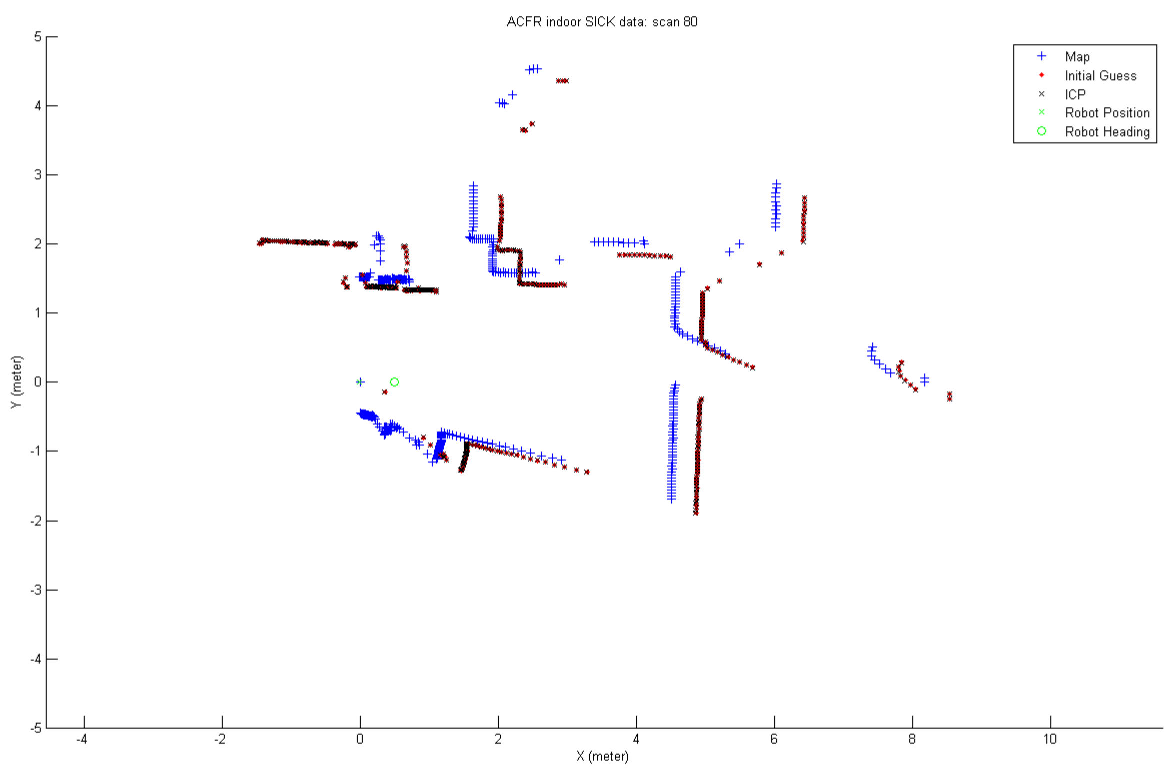
Incorporating the ICPv4.m algorithm into the laserShowACFR.m program enables us to build a real time picture of the movement of the vehicle and its perceived surroundings, relative to the actual mapped data. Observe:

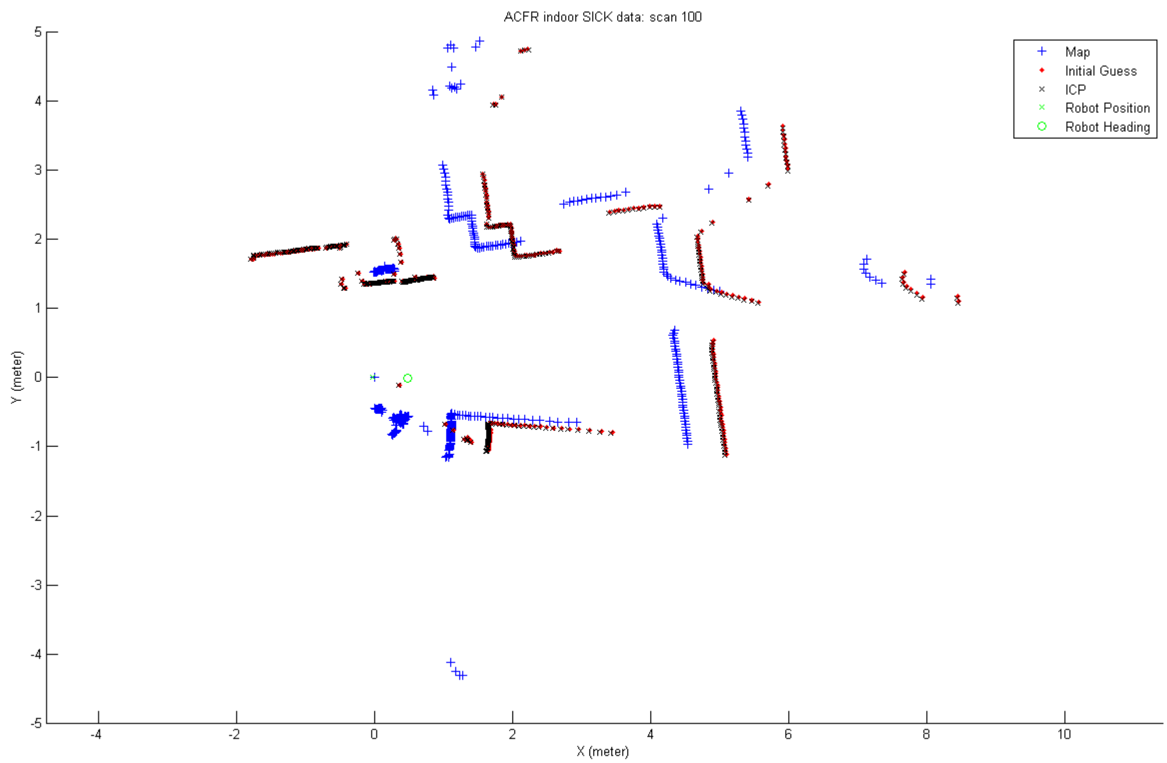


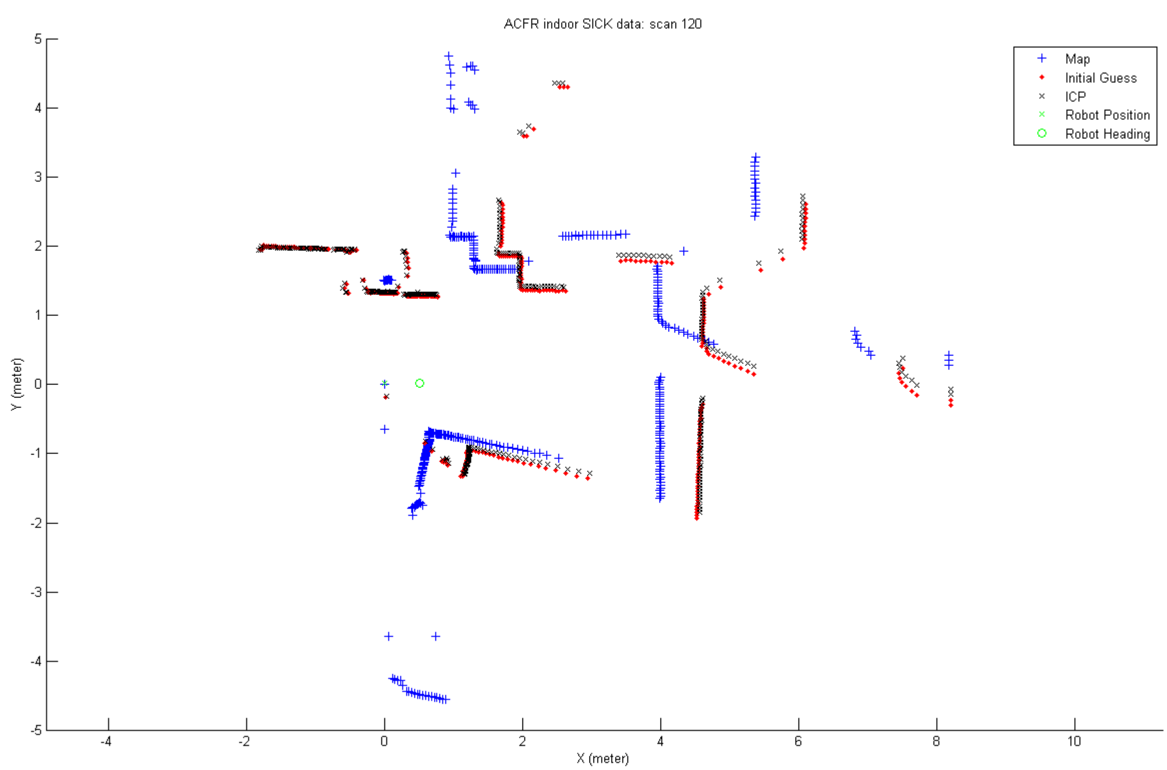

ACFR indoor SICK data: scan 20
Robot Position
Robot Heading
Map
Initial Guess
ICP
Y (meter)
X (meter)

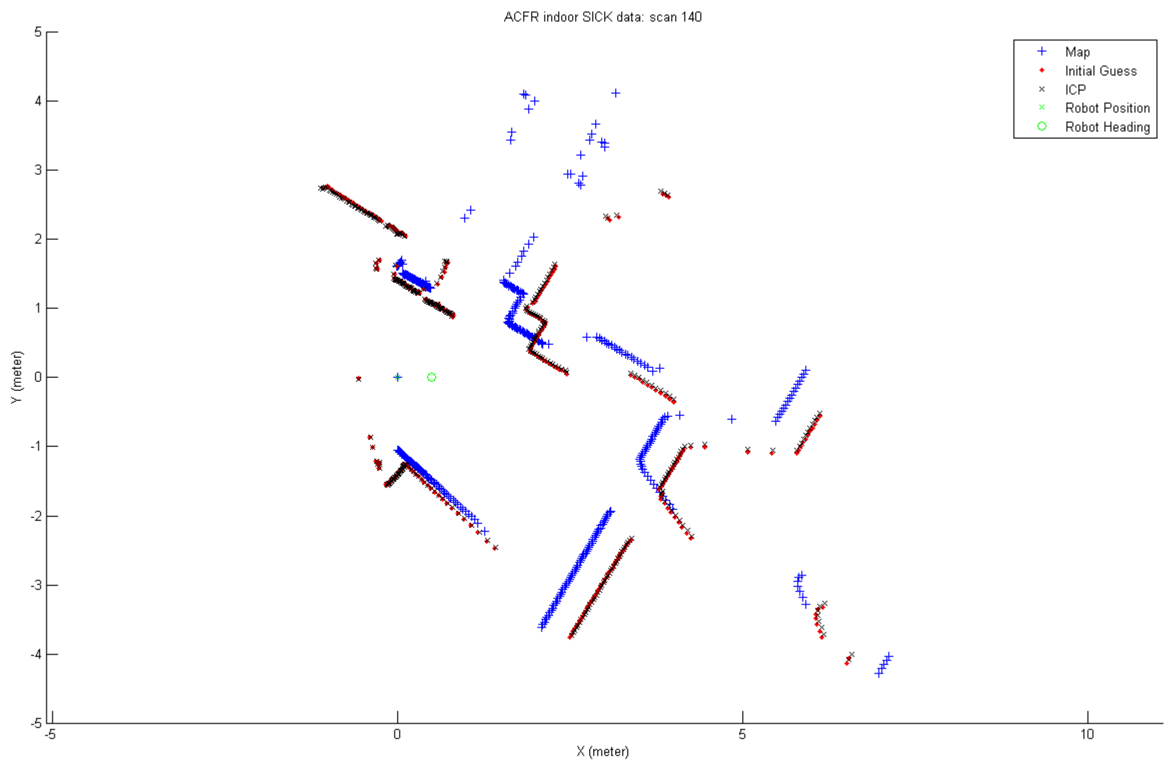

ACFR indoor SICK data: scan 40
Robot Position
Robot Heading
Map
Initial Guess
ICP
Y (meter)
X (meter)

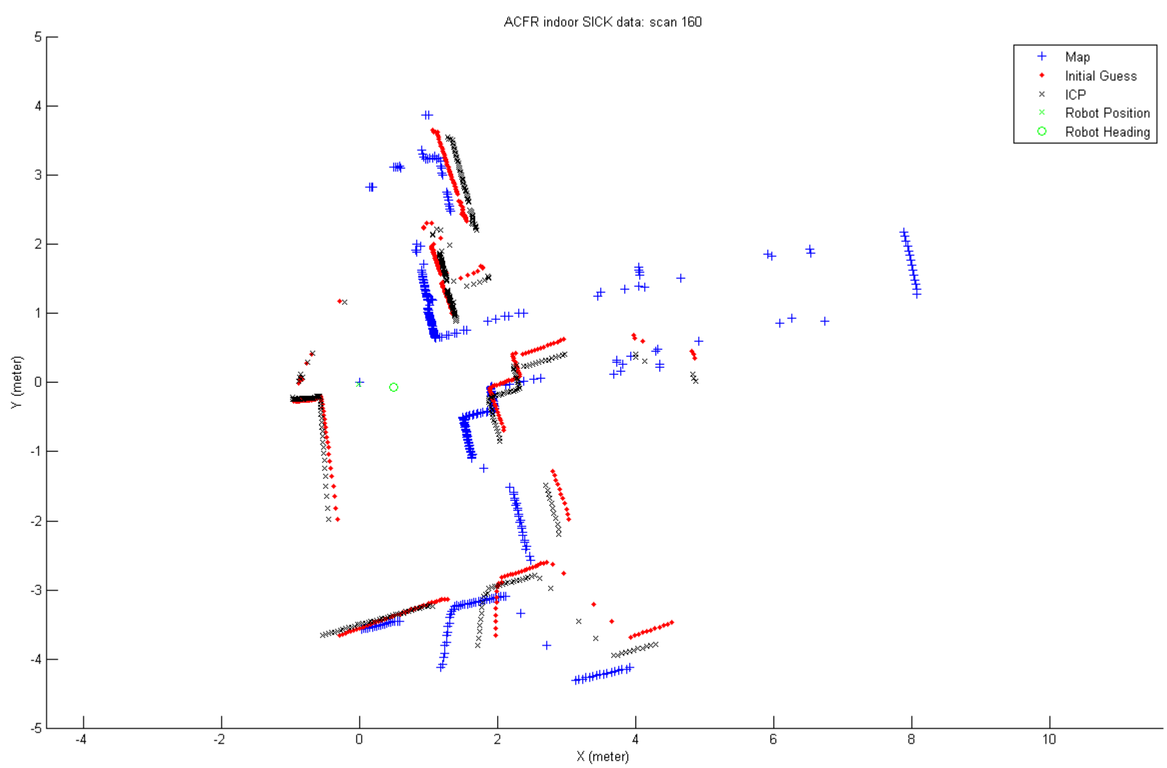

ACFR indoor SICK data: scan 60
Robot Position
Robot Heading
Map
Initial Guess
ICP
X (meter)
Y (meter)

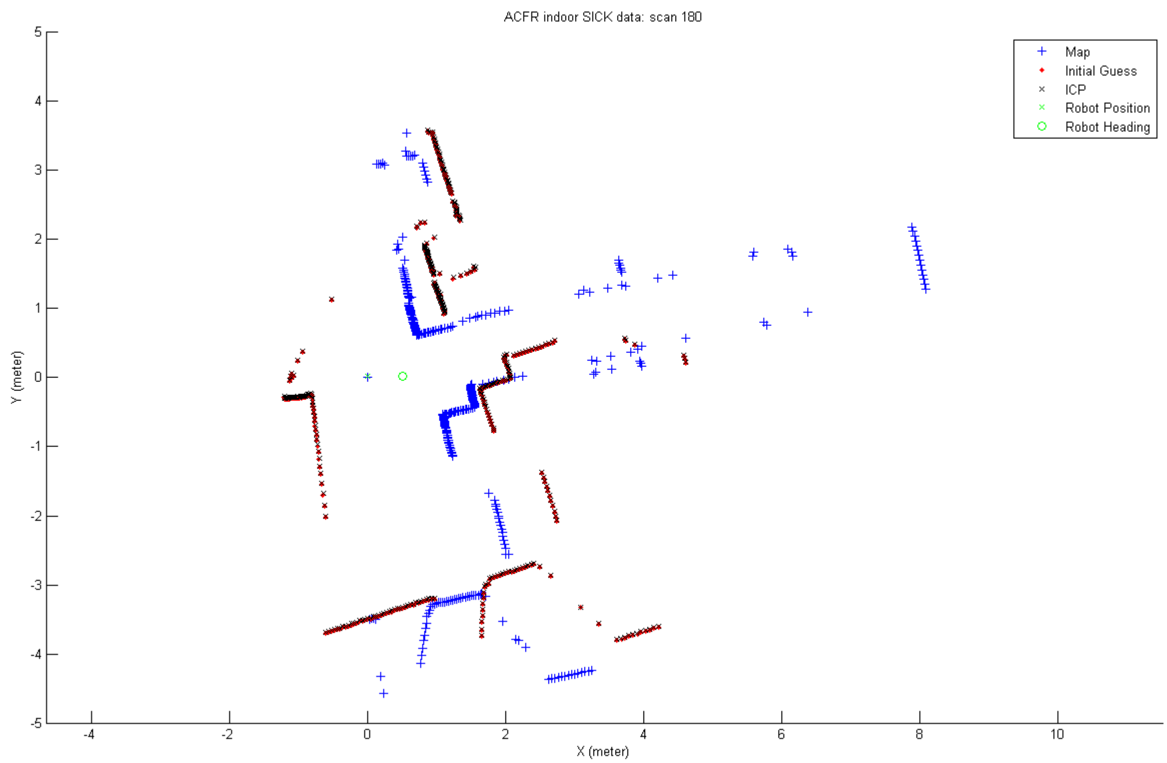

ACFR indoor SICK data: scan 80
Map
Initial Guess
ICP
Robot Position
Robot Heading
X (meter)
Y (meter)

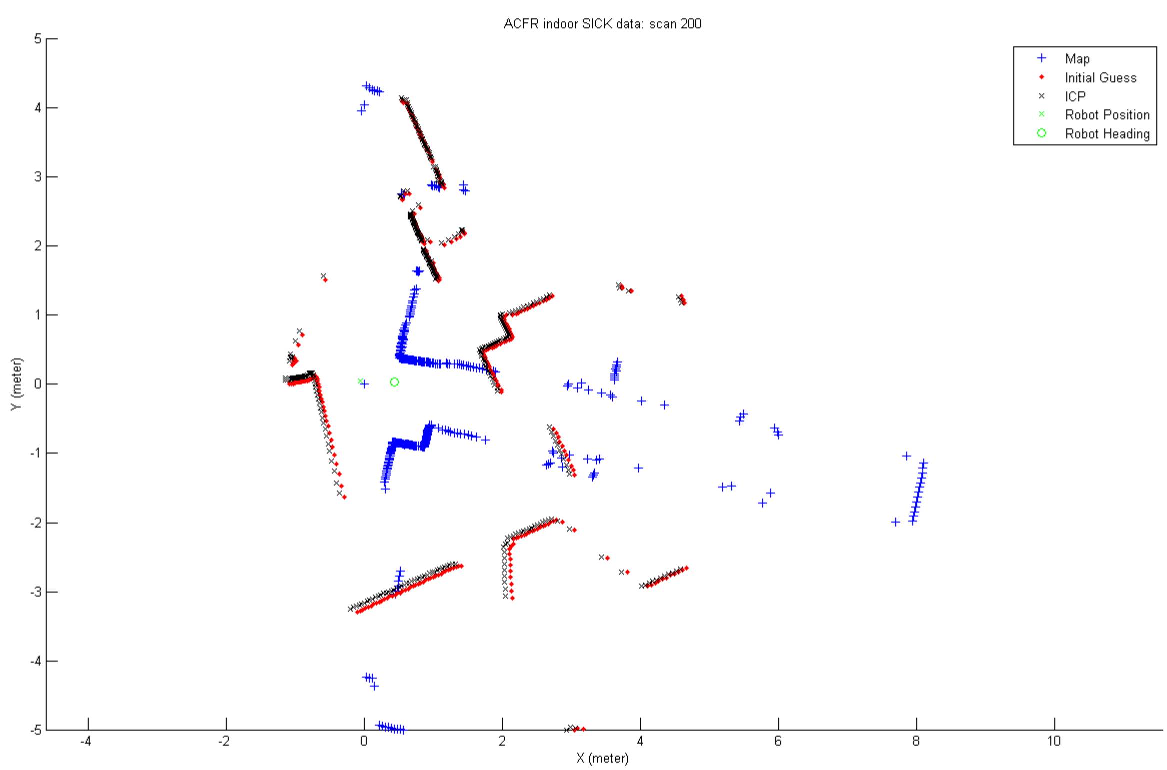

ACFR indoor SICK data: scan 100
Map
Initial Guess
ICP
Robot Position
Robot Heading
Y (meter)
X (meter)

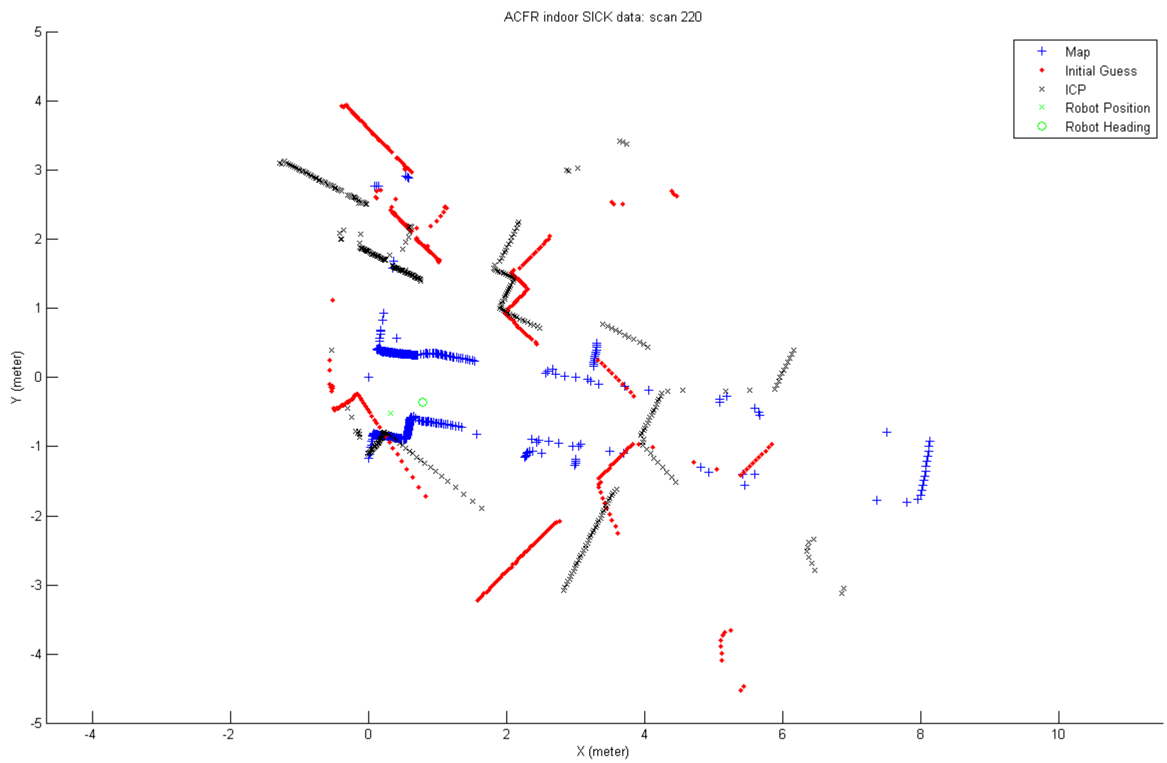

ACFR indoor SICK data: scan 120
Map
Initial Guess
ICP
Robot Position
Robot Heading
Y (meter)
X (meter)

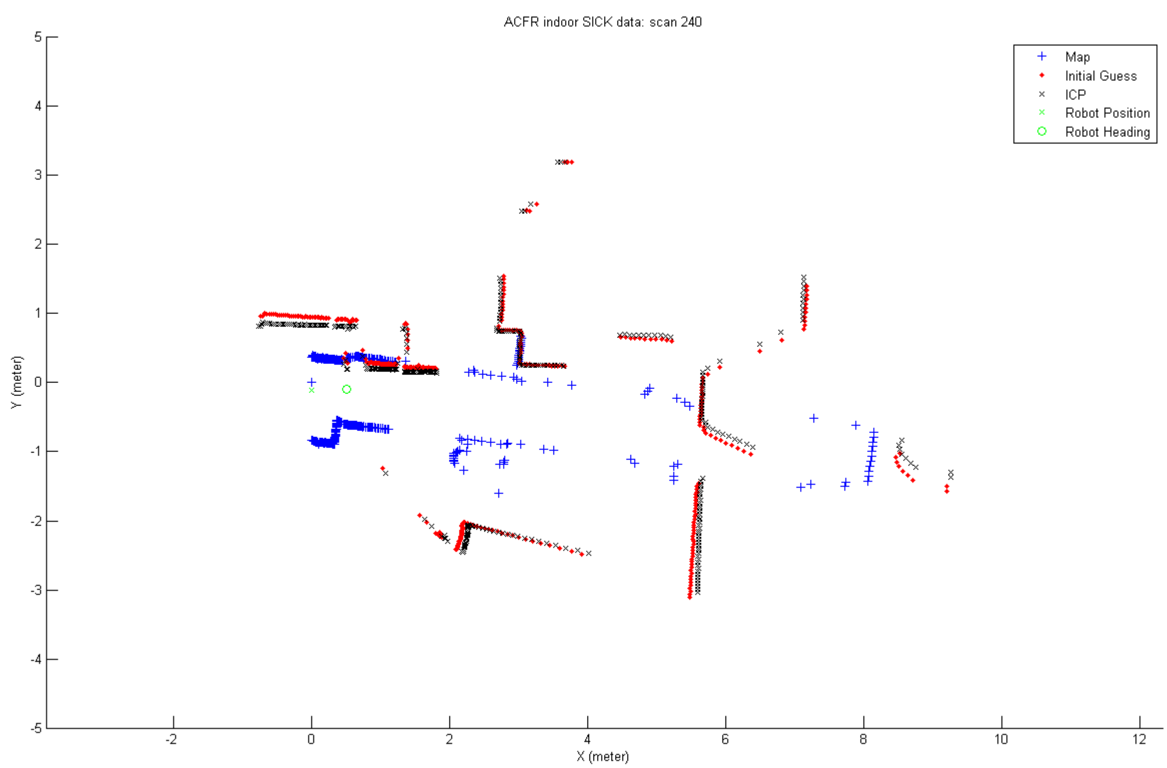

ACFR indoor SICK data: scan 140
Map
Initial Guess
ICP
Robot Position
Robot Heading
Y (meter)
X (meter)

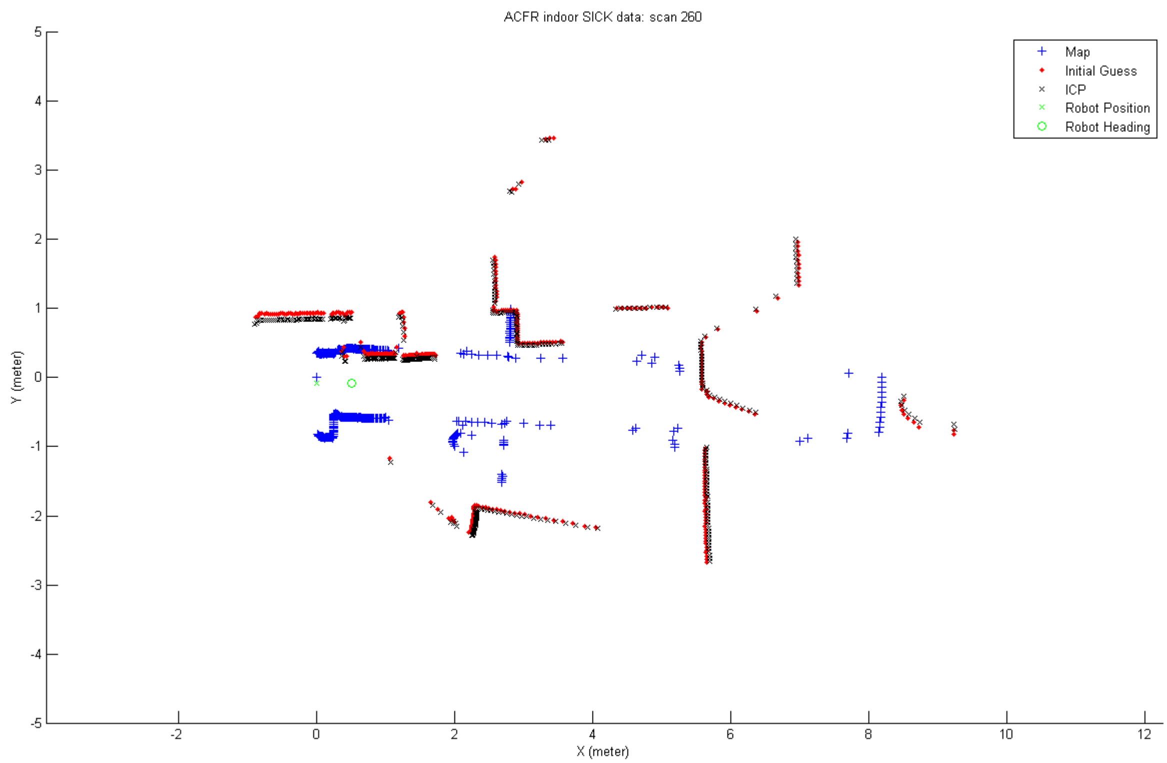

ACFR indoor SICK data: scan 160
Map
Initial Guess
ICP
Robot Position
Robot Heading
Y (meter)
X (meter)

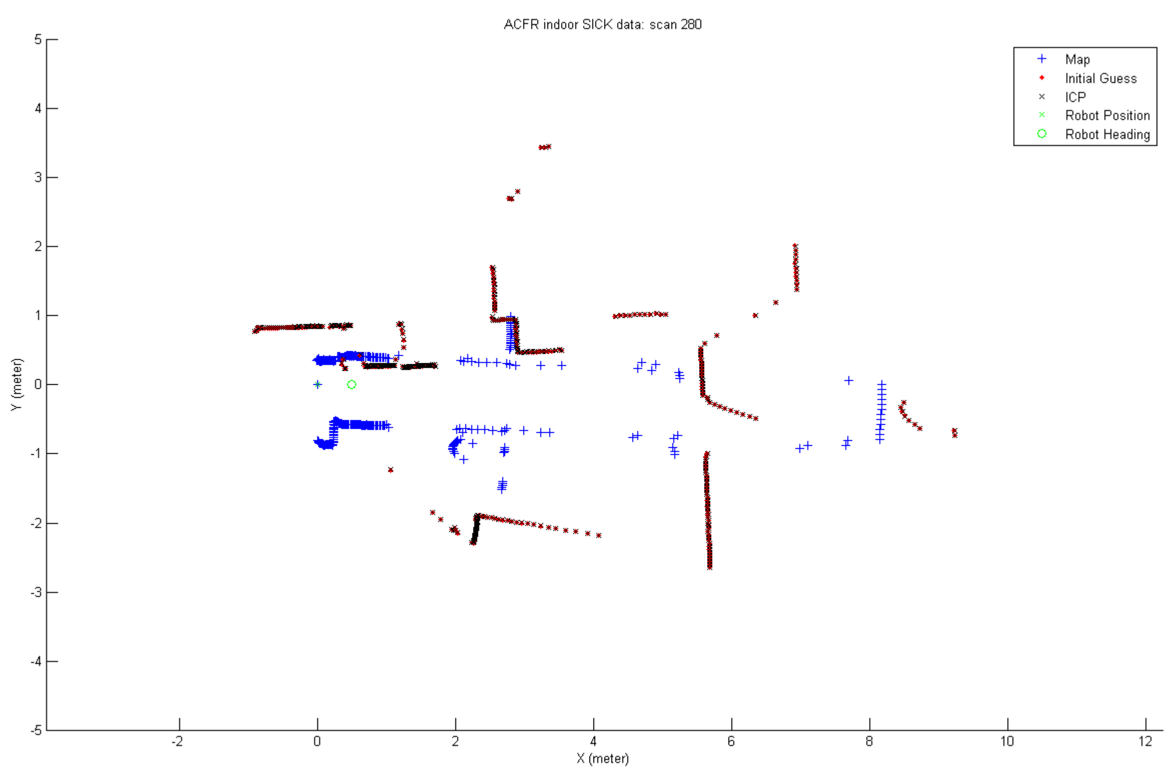

ACFR indoor SICK data: scan 180
Map
Initial Guess
ICP
Robot Position
Robot Heading
Y (meter)
X (meter)

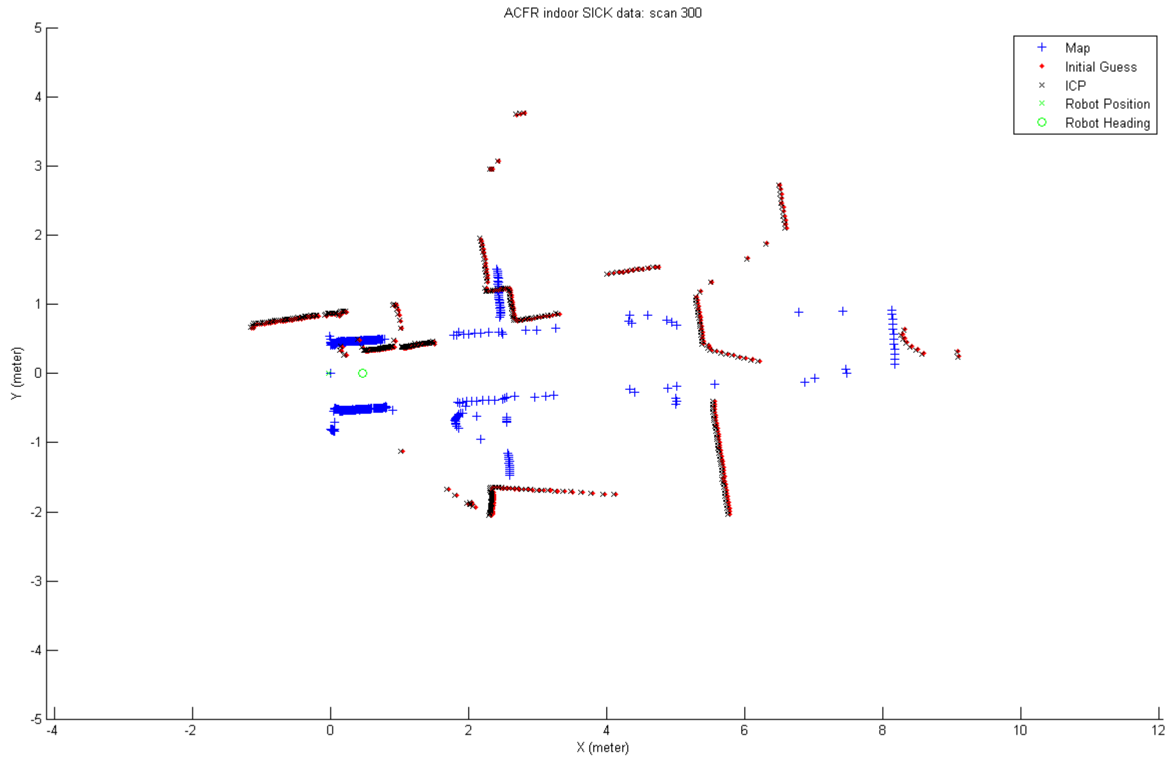

ACFR indoor SICK data: scan 200
Map
Initial Guess
ICP
Robot Position
Robot Heading
Y (meter)
X (meter)


ACFR indoor SICK data: scan 220
Map
Initial Guess
ICP
Robot Position
Robot Heading
Y (meter)
X (meter)
5
4
3
2
1
0
-1
-2
-3
-4
-5
-4
-2
0
2
4
6
8
10


ACFR indoor SICK data: scan 240
Map
Initial Guess
ICP
Robot Position
Robot Heading
Y (meter)
X (meter)
5
4
3
2
1
0
-1
-2
-3
-4
-5
-2
0
2
4
6
8
10
12


ACFR indoor SICK data: scan 260
Map
Initial Guess
ICP
Robot Position
Robot Heading
Y (meter)
X (meter)
5
4
3
2
1
0
-1
-2
-3
-4
-5
-2
0
2
4
6
8
10
12


ACFR indoor SICK data: scan 280
Map
Initial Guess
ICP
Robot Position
Robot Heading
Y (meter)
X (meter)
5
4
3
2
1
0
-1
-2
-3
-4
-5
-2
0
2
4
6
8
10
12



As can be seen, the ICP initially tracks the map data rather well, and the vehicle remains roughly alligned with the direction of motion. Around scan 200 however, everything begins to break down. At this point, the ICP has started lagging behind, and when the vehicle suddenly rotates it is unable to keep up. The closest points now no longer correspond to the original positions, and the ICP map becomes distorted, aligining with a new set of map data points. By scan 300 the alignment is well an truly distorted, with the ICP data almost backwards on the map data. The scans remain relatively unchanged for the next thousand frames, which have not been shown here. The vehicle, suprisingly, appears to be orientated correctly. This is probably a fluke.

Considering that at some points in the scans the ICP vehicle position placed it in or beyond the walls, this would not be a particularly effect method of guidance.

Note, however, that though there was always some offset between the ICP map and the real map, this offset was almost constant. The greatest offset would occur during rotations, which the ICP was able to follow initially. However, these first few rotations where slow, enabling the ICP algorithm to keep up. Around frame 200, the first rapid rotation took place, and the ICP finally failed to keep up, recognising new points as the closest and pairing with them.
As mentioned above, this is because the ICP algorithm fails to take this rotational element into account. However, the given algorithm would still be acceptable under the correct circumstances. With either far more data points taken per seconds, enabling the ICP to 'linearise' the rotations, would enable it to handle sudden rapid rotations. Alternatively, the vehicle could be drastically slowed down, resulting in more data points through the rotation, again enabling some degree of linearity.

Code Listing

See Appendix A [3] for all code used.

4 Question 4

4.a Modified DH Notation

5 Question 5

5.a

5.a.i Workspace

5.b Singularities

6 Question 6

6.a Inverse Kinematics

6.a.i Theoretical Method

6.a.ii Results

6.a.iii Code Listing

See Appendix A [9.5]

6.b Simulation

7 Appendix A

7.1 Question 1

7.2 Question 2

7.3 Question 3

7.3.i showICP

```
% test the ICP algorithm
% Author: Chieh-Chih (Bob) Wang [bob.wang@cas.edu.au]
% Created: April 12, 2005.
% Last Modified: April 12, 2005.


clear
close all
clc

% load laser files
laser_scans=load('datasets\captureScanshornet.txt');
t0 = laser_scans(1,1);

% Scan A
i = 500;
xA = zeros(1);
yA = zeros(1);
for j = 2:size(laser_scans,2)   %Map
 range = laser_scans(i,j) / 1000;
 bearing = ((j-1)/2 - 90)*pi/180;
 if (range < 75)
     xA = [xA range*cos(bearing)];
     yA = [yA range*sin(bearing)];
 end
end

% Scan B
i = 520;
xB = zeros(1);
yB = zeros(1);
for j = 2:size(laser_scans,2)   %Initial Guess (source? stay 'constant'? trasformed by ICP)
 range = laser_scans(i,j) / 1000;
 bearing = ((j-1)/2 - 90)*pi/180;
 if (range < 75)
     xB = [xB range*cos(bearing)];
     yB = [yB range*sin(bearing)];
 end
end

deltaPose = zeros(3,1);

[deltaPose_bar, deltaPose_bar_Cov, N] = ICPv4(deltaPose, [xA;yA], [xB;yB]); %ICP

newB = head2tail_no_theta(deltaPose_bar, [xB;yB]);
new_xB = newB(1,:);
new_yB = newB(2,:);

figure
clf
hold on
plot(xA,yA,'b+')
plot(xB,yB,'r.')
plot(new_xB,new_yB,'kx')
axis equal
legend('Map','initial guess','ICP')
xlabel('X (meter)')
ylabel('Y (meter)')
title('The ICP algorithm')
```

7.3.ii Modified showICP

```
% test the ICP algorithm
% Author: Chieh-Chih (Bob) Wang [bob.wang@cas.edu.au]
% Created: April 12, 2005.
% Last Modified: April 12, 2005.
%
%Modified by James Ferris to Show ICP data in a plot


clear
close all
clc

% load laser files
laser_scans=load('datasets\captureScanshornet.txt');
t0 = laser_scans(1,1);

% Scan A
i = 500;
xA = zeros(1);
yA = zeros(1);
for j = 2:size(laser_scans,2)   %Map
 range = laser_scans(i,j) / 1000;
 bearing = ((j-1)/2 - 90)*pi/180;
 if (range < 75)
     xA = [xA range*cos(bearing)];
     yA = [yA range*sin(bearing)];
 end
end

% Scan B
i = 520;
xB = zeros(1);
yB = zeros(1);
for j = 2:size(laser_scans,2)   %Initial Guess (source? stay 'constant'? trasformed by ICP)
 range = laser_scans(i,j) / 1000;
 bearing = ((j-1)/2 - 90)*pi/180;
 if (range < 75)
     xB = [xB range*cos(bearing)];
     yB = [yB range*sin(bearing)];
 end
end

deltaPose = zeros(3,1);

[deltaPose_bar, deltaPose_bar_Cov, N] = ICPv4(deltaPose, [xA;yA], [xB;yB]); %ICP

%%
%Added to show robot position and orientation
figure
clf
hold on
Pose = deltaPose_bar;
h = 0.5;
Pose2 = [Pose(1)+h*cos(Pose(3)), Pose(2)+h*sin(Pose(3))];
plot(Pose(1),Pose(2),'gx');
plot(Pose2(1),Pose2(2),'go');
%%

newB = head2tail_no_theta(deltaPose_bar, [xB;yB]);
new_xB = newB(1,:);
new_yB = newB(2,:);

plot(xA,yA,'b+')
plot(xB,yB,'r.')
plot(new_xB,new_yB,'kx')
axis equal
legend('Robot Position', 'Robot Heading', 'Map','Initial Guess','ICP')
```

```
xlabel('X (meter)')
ylabel('Y (meter)')
title('The ICP algorithm')
```

7.3.iii laserShowAcfr

```
%
% Author: Stefan Williams (stefanw@acfr.usyd.edu.au)
%
%

clear
close all
clc

% load laser files
laser_scans=load('datasets\captureScanshornet.txt');

t0 = laser_scans(1,1);

figure
for i = 1:length(laser_scans)
     tlaser = laser_scans(i,1) - t0;
     xpoint = zeros(1);
     ypoint = zeros(1);
     for j = 2:size(laser_scans,2)
         range = laser_scans(i,j) / 1000;
         bearing = ((j-1)/2 - 90)*pi/180;
         if (range < 75)
             xpoint = [xpoint range*cos(bearing)];
             ypoint = [ypoint range*sin(bearing)];
         end
     end
     plot(xpoint(:), ypoint(:), '.');
     axis equal;
     axis([0 10 -5 5]);
     xlabel('X (meter)')
     ylabel('Y (meter)')
     title(sprintf('ACFR indoor SICK data: scan %d',i))
     drawnow

%      if i == 500
%          pause;
%      elseif i == 520
%              pause;
%      end

end
```

7.3.iv Modified laserShowACFR

```
%
% Author: Stefan Williams (stefanw@acfr.usyd.edu.au)
%
%
%Modified by James Ferris to include ICP data in a plot

clear
close all
clc

% load laser files
laser_scans=load('datasets\captureScanshornet.txt');

t0 = laser_scans(1,1);

%%
%Initialise ICP data
i = 20; %Let the 'initial guess' be starting point + 20, as in part A
xB = zeros(1);
yB = zeros(1);
for j = 2:size(laser_scans,2)
    range = laser_scans(i,j) / 1000;
    bearing = ((j-1)/2 - 90)*pi/180;
    if (range < 75)
      xB = [xB range*cos(bearing)];
      yB = [yB range*sin(bearing)];
    end
end
deltaPose = zeros(3,1);

%%

figure
for i = 1:length(laser_scans)

     tlaser = laser_scans(i,1) - t0;
     xpoint = zeros(1);
     ypoint = zeros(1);
     for j = 2:size(laser_scans,2)
         range = laser_scans(i,j) / 1000;
         bearing = ((j-1)/2 - 90)*pi/180;
         if (range < 75)
             xpoint = [xpoint range*cos(bearing)];
             ypoint = [ypoint range*sin(bearing)];
         end
     end

     %%
     %Calculate ICP
     [deltaPose_bar, deltaPose_bar_Cov, N] = ICPv4(deltaPose, [xpoint;ypoint], [xB;yB]);
     newB = head2tail_no_theta(deltaPose_bar, [xB;yB]);
     new_xB = newB(1,:);
     new_yB = newB(2,:);

     if i == 1 || mod(i,20)== 0

         Pose = deltaPose_bar;
         h = 0.5;
         Pose2 = [Pose(1)+h*cos(Pose(3)), Pose(2)+h*sin(Pose(3))];
         hold on


         plot(xpoint(:), ypoint(:), '.');

         plot(xB,yB,'r.')
         plot(new_xB,new_yB,'kx')

```

```
        plot(Pose(1),Pose(2),'gx');
        plot(Pose2(1),Pose2(2),'go');


        %%


        axis equal;

        legend('Robot Position', 'Robot Heading', 'Map','Initial Guess','ICP')
        axis([0 10 -5 5]);
        xlabel('X (meter)')
        ylabel('Y (meter)')
        title(sprintf('ACFR indoor SICK data: scan %d',i))
        drawnow

        pause
    end

    xB = new_xB;
    yB = new_yB;

    clf

end
```

7.3.v ICPv4

```
function [deltaPose_bar, deltaPose_bar_Cov, N] = ICPv4(deltaPose, a, b)

% function: ICP algorithm version 3.0
% a: point set, 2 x Na
% b: point set, 2 x Nb
% Xab: the relative tranformation between a and b
%
%
%
% Author: Chieh-Chih (Bob) Wang [bobwang@cs.cmu.edu]
% Created: Dec. 2, 2002.
% Last Modified: Dec. 27, 2003.
% (C) 2002-2004 Chieh-Chih (Bob) Wang. All Rights Reserved.


max_delta_g = 0.01;
max_iter = 40;
WinSize = 80;

% Cell 5cm x 5cm
grid_size =   0.005; % Changing this value may affect the accuracy of the ICP algorithm.
Map_x_min =   min(a(1,:));
Map_y_min =   min(a(2,:));
Map_x_max =   max(a(1,:));
Map_y_max =   max(a(2,:));

% Step 1: Create a grid map foo speed up correspondence search
GridMap_X = ceil((Map_x_max - Map_x_min)/grid_size);
GridMap_Y = ceil((Map_y_max - Map_y_min)/grid_size);
GridMap = zeros(GridMap_X, GridMap_Y);

for k=1:size(a,2)
    [Map_i,Map_j] = XY2IJ(a(1,k), a(2,k), grid_size, Map_x_min, Map_y_min);
    if Map_i>0 & Map_i<= GridMap_X & Map_j>0 & Map_j<= GridMap_Y
        if GridMap(Map_i,Map_j)~= 0
            %disp(sprintf('points collide %d,%d', Map_i, Map_j))
        end
        GridMap(Map_i,Map_j) = k;
    end
end

WinSize_org = WinSize;
delta_g = 1000000000;
j=0;
g = deltaPose;

% method 2:
Z = [];
M = [];

NoMatch_flag = 0;

while (j < max_iter) & (delta_g > max_delta_g)
    j = j+1;
    old_g = g;
    % Step 1:

    % Finding Correspondence
    Match_Pairs = [];
    %New_Scan1_Index = Scan1_Index;
    %New_Scan2_Index = Scan2_Index;

    %WinSize = round(WinSize_org/j);
    WinSize = WinSize_org- 4*j;
    if WinSize < 2
        WinSize = 2;
    end
```

```
for k=1:size(b,2)

    Point_X = head2tail_no_theta(g,b(:,k));

    [Map_i,Map_j] = XY2IJ(Point_X(1), Point_X(2), grid_size, Map_x_min, Map_y_min);
    % Search ...
    % Define search area
    % WinSize = 10;
    Win_i_min = Map_i - WinSize;
    if Win_i_min < 1
        Win_i_min = 1;
    end
    Win_i_max = Map_i + WinSize;
    if Win_i_max > GridMap_X
        Win_i_max = GridMap_X;
    end
    Win_j_min = Map_j - WinSize;
    if Win_j_min < 1
        Win_j_min = 1;
    end
    Win_j_max = Map_j + WinSize;
    if Win_j_max > GridMap_Y
        Win_j_max = GridMap_Y;
    end
    [Search_i, Search_j] = find(GridMap(Win_i_min:Win_i_max, Win_j_min:Win_j_max) > 0);
    if size(Search_i,1)>0
        min_dis = 100000000000000;
        match_index = 0;
        for m=1:size(Search_i,1)
            a_index = GridMap(Search_i(m)+Win_i_min-1, Search_j(m)+Win_j_min-1);
            dis = sqrt((Point_X(1) - a(1,a_index))^2 ...
                + (Point_X(2) - a(2,a_index))^2);
            if (dis < min_dis)
                min_dis = dis;
                match_index = a_index;
            end
        end
        % method 1:
        Match_Pairs = [Match_Pairs; ...
                b(1,k) b(2,k) a(1,match_index) a(2,match_index)];
        % method 2:
        Mk = [1 0 -Point_X(2,1); 0 1 Point_X(1,1)];
        M = [M; Mk];
        Z = [Z; Point_X - a(:,match_index)];

        %MatchedPoints(1,k) = 1;
        %New_Scan1_Index(1, match_index) = 2; % see Readme.txt for the definition
        %New_Scan2_Index(1, k) = 2;
    end
end
%Method 1: the closed form solution without covariance estimate
N = size(Match_Pairs,1);
%disp(sprintf('Inside ICPv4: iter %d, match pairs %d',j, N));
if N == 0
    NoMatch_flag = 1;
    break
end

X2_bar = sum(Match_Pairs(:,1))/N;
Y2_bar = sum(Match_Pairs(:,2))/N;
X1_bar = sum(Match_Pairs(:,3))/N;
Y1_bar = sum(Match_Pairs(:,4))/N;
Sx2x1 = sum((Match_Pairs(:,1) - X2_bar).*(Match_Pairs(:,3) - X1_bar));
Sy2y1 = sum((Match_Pairs(:,2) - Y2_bar).*(Match_Pairs(:,4) - Y1_bar));
Sx2y1 = sum((Match_Pairs(:,1) - X2_bar).*(Match_Pairs(:,4) - Y1_bar));
Sy2x1 = sum((Match_Pairs(:,2) - Y2_bar).*(Match_Pairs(:,3) - X1_bar));

g(3,1) = atan2(Sx2y1-Sy2x1, Sx2x1+ Sy2y1);
g(1,1) = X1_bar - (X2_bar*cos(g(3,1))- Y2_bar*sin(g(3,1)));
g(2,1) = Y1_bar - (X2_bar*sin(g(3,1))+ Y2_bar*cos(g(3,1)));
```

```
    delta_g = sqrt((old_g(1) - g(1))^2 + (old_g(2) - g(2))^2);

    %Method 2:
%     InvMM = inv(M'*M);
%     D_bar = InvMM*M'*Z;
%     ZminusMD_bar =Z-M*D_bar;
%     s_square = ZminusMD_bar'*ZminusMD_bar/(2*N-3);
%     Cov_ICP = s_square*InvMM;

    %pause
end
% Xab_bar = g+D_bar;
% Xab_Cov = Cov_ICP;
if NoMatch_flag == 0
    InvMM = inv(M'*M);
    D_bar = InvMM*M'*Z;
    ZminusMD_bar =Z-M*D_bar;
    s_square = ZminusMD_bar'*ZminusMD_bar/(2*N-3);
    Cov_ICP = s_square*InvMM;

    deltaPose_bar = g;
    deltaPose_bar_Cov = Cov_ICP;
else
    deltaPose_bar = [];
    deltaPose_bar_Cov = [];
end
```

7.3.vi XY2IJ

```
function [Map_i,Map_j] = XY2IJ(x, y, grid_size, Map_x_min, Map_y_min)

% XY2IJ
%
% Author: Chieh-Chih (Bob) Wang [bobwang@cs.cmu.edu]
% Created: Oct. 31, 2002.
% Modified: Nov. 6, 2003.
% (c) 2002-2003 Chieh-Chih Wang. All Rights Reserved.

% Changed to the vec form
Map_i = round((x - Map_x_min)/grid_size + 0.5);
Map_j = round((y - Map_y_min)/grid_size + 0.5);
```

7.3.vii head2tail no theta

```
function Xik = head2tail_no_theta(Xij, Xjk)

% Compounding Operation: head2tail
%
% Input:
%   Xij = [x_ij; y_ij; theta_ij]
%   Xjk = [x_jk; y_jk]
% Output:
%   Xik = [x_ik; y_ik]
%
% Author: Chieh-Chih (Bob) Wang [bobwang@cs.cmu.edu]
% Created: Nov. 8, 2002.
% Modified: Nov. 6, 2003.


cosTheta_ij = cos(Xij(3,1));
sinTheta_ij = sin(Xij(3,1));

% Xik(1,1) = Xjk(1,1)*cosTheta_ij - Xjk(2,1)*sinTheta_ij + Xij(1,1);
% Xik(2,1) = Xjk(1,1)*sinTheta_ij + Xjk(2,1)*cosTheta_ij + Xij(2,1);

% Change for Vec...
Xik(1,:) = Xjk(1,:)*cosTheta_ij - Xjk(2,:)*sinTheta_ij + Xij(1,1);
Xik(2,:) = Xjk(1,:)*sinTheta_ij + Xjk(2,:)*cosTheta_ij + Xij(2,1);
```

7.4 Question 4

7.5 Question 5

7.6 Question 6