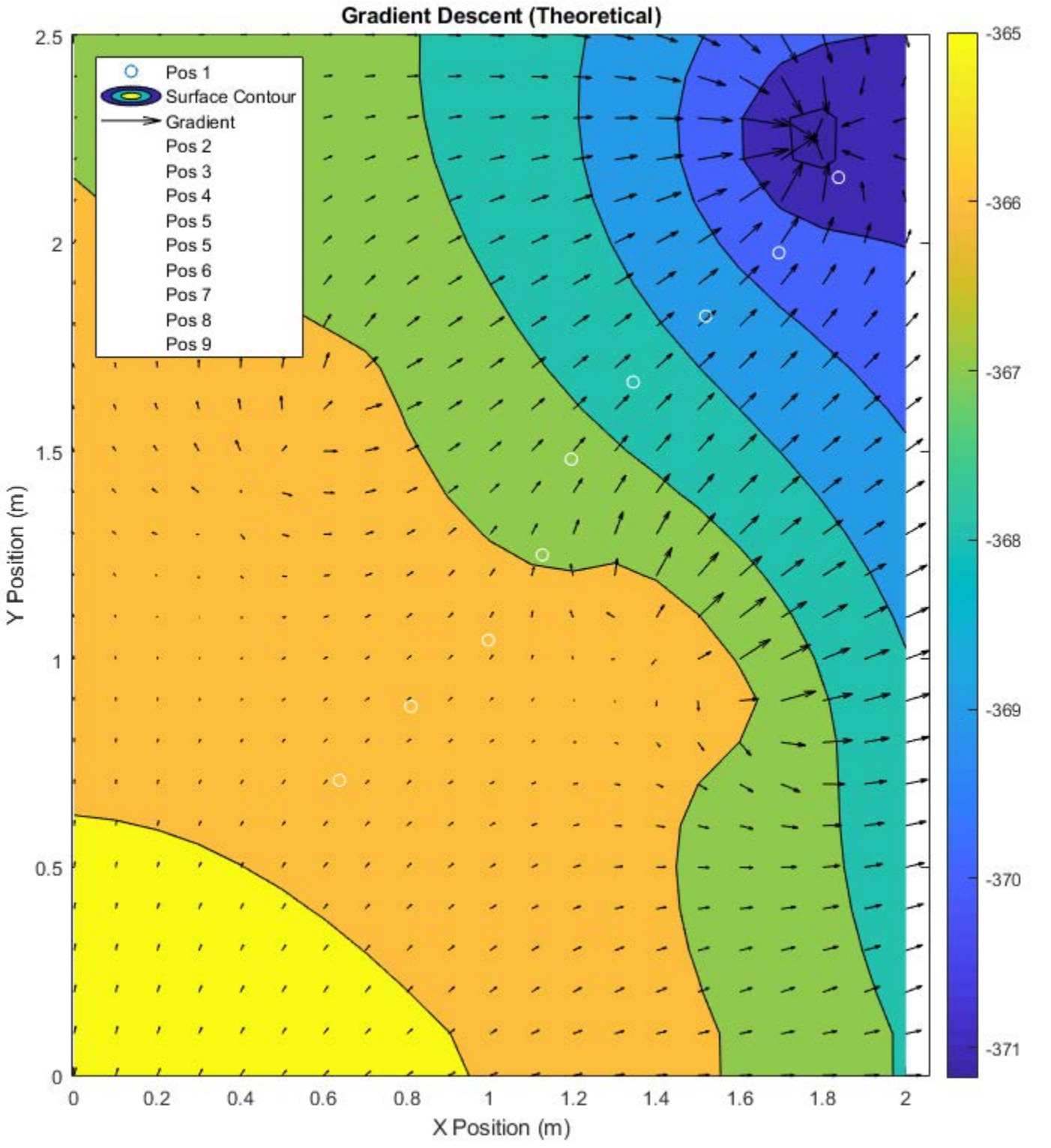
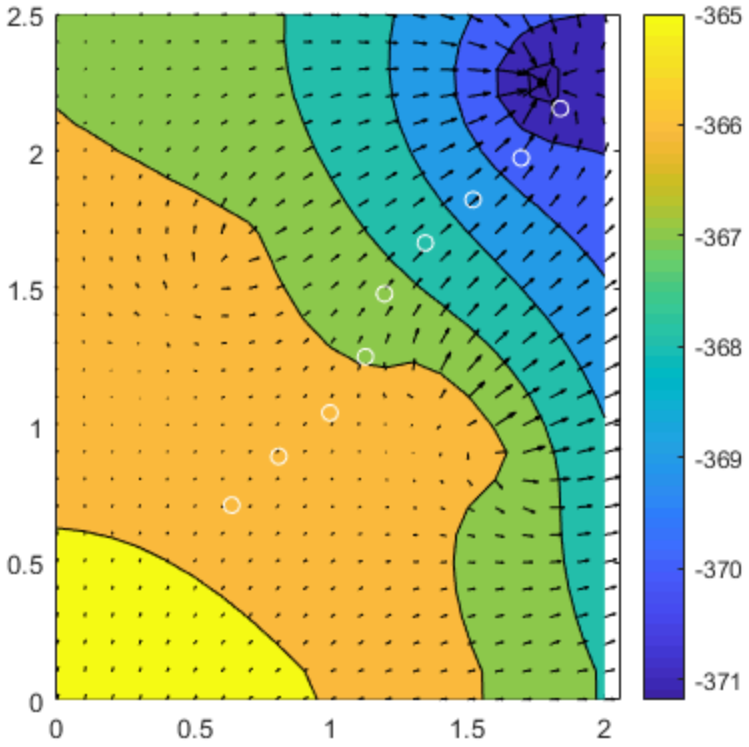

Gradient Descent (Theoretical)
Pos 1
Surface Contour
Gradient
Pos 2
Pos 3
Pos 4
Pos 5
Pos 5
Pos 6
Pos 7
Pos 8
Pos 9
Y Position (m)
X Position (m)
2.5
2
1.5
1
0.5
0
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
-365
-366
-367
-368
-369
-370
-371

```
figure(1), clf

tic
% choose initial point
r = [0.5;0.5];

% generate the surface with lines and such
plot(r(1), r(2),'o')
hold on

[px,py]=meshgrid(0:0.1:2,0:0.1:2.5);
[xlim,ylim] = size(px);
V = zeros(xlim, ylim);

for i=1:xlim
    for j=1:ylim
          line = @(x0,m,xi,yi) log(sqrt(( px(i,j)-x0-xi).^2 +
 ((py(i,j)-(m.*x0)-yi).^2)));
          circle = @(xi,yi) log(sqrt((px(i,j)-xi).^2 + ((py(i,j)-
yi).^2)));
          dV4 = @(x0)line(x0, -1,1.5,1);
          dV5 = @(x0)line(x0, 1,0.5,1.5);
          dV3 = @(x0)line(x0, 0.01,0,0);
          dV6 = @(x0)line(x0, 0.01,0,2.5);
          dV7 = @(x0)line(x0, -1,0,0);
          dV8 = @(x0)line(x0, 1000,2,0);
          V(i,j) = integral(dV5,0.2,-0.2) + 2*integral(dV4,0.2,-0.2)
 + integral(dV3,4,-4) + integral(dV6,4,-4) + integral(dV7,4,-4)./3 +
 6*integral(dV8,4,-4) + circle(1.75,2.25);
    end
end

% visualize contour and gradient plot of surface in advance
% (entirely for pretty plots)
contourf(px,py,V);
[Ex,Ey] = gradient(V);
axis equal
hold on
colorbar
quiver(px,py,-Ex,-Ey,'k');

% find the gradient at initial pt
xp = round(r(1),2);
yp = round(r(2),2);
[gx, gy] = gradGenerator(xp,yp);
grad = 30.*[-gx;-gy]


lambda = .25; % feet
delta = .99; % current delta
tolerance = .1; % gradient norm tolerance
orientation = [0;1]; %initial orientation
```

```
% calculate desired angle
angle = acos(dot(orientation, grad)./norm(grad));
orientation = grad./norm(grad);
time = 2*angle/(pi / 2);

norm(grad);
count = 1;

disp('gradient time')

while norm(grad) > tolerance
    if count < 10
        time = (lambda/3.281)/0.1;

        % update gradient, linear, and angle calculations
        r = r + lambda*grad./norm(grad);
        lambda = lambda*delta;
        xp = round(r(1),2);
        yp = round(r(2),2);
        [gx, gy] = gradGenerator(xp,yp);
        grad = 30.*[-gx;-gy];
        norm(grad);
        angle = acos(dot(orientation, grad)./norm(grad));
        orientation = grad./norm(grad);

        time = 2*angle/1.5748;

        plot(r(1), r(2),'wo');
        count = count+1;
    else
        break
    end
end

toc


grad =

    0.1657
    0.2517

gradient time
Elapsed time is 1.867181 seconds.
```




*Published with MATLAB® R2019a*

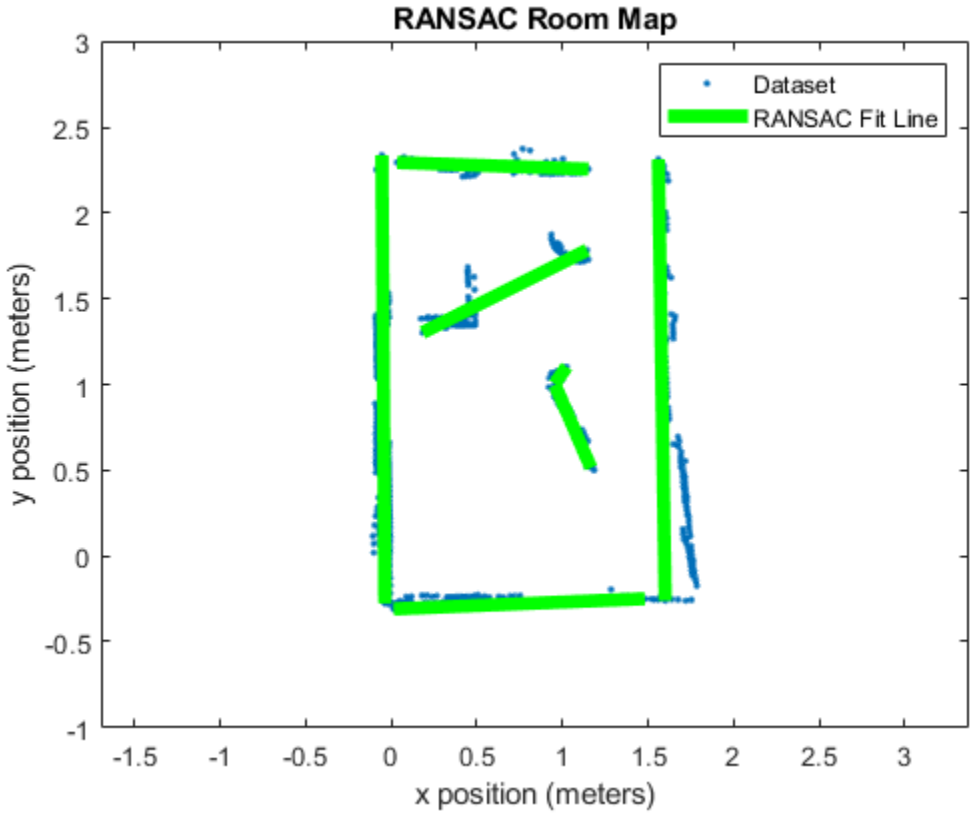
# RANSAC Implementation

```
clear;rng(3);
load('allRoomDataCleaned.mat');
x = data(1,:)';
y = data(2,:)';
figure(1);clf;
plot(x,y, '.');hold on;
axis equal;
title('RANSAC Room Map');
xlabel('x position (meters)');
ylabel('y position (meters)');

clearvars -except x y
tic
[A, B, bestTestLine, outliers, inliers] = RANSAC(x,y,0.05,100);

% plot(inliers(:,1), inliers(:,2), 'ks')

p = polyfit([A(1) B(1)], [A(2) B(2)], 1);
% testline = polyval(p, x);

lengths = vecnorm(inliers' - [10; 10]);
[~, ia] = min(lengths);
[~, ib] = max(lengths);
end1 = inliers(ia,:);
end2 = inliers(ib,:);
line = [end1; end2];
plot(line(:,1), line(:,2), 'g', 'LineWidth', 5)


% plot(x, testline, 'ro')

for i = 1:6
    x = outliers(:,1);
    y = outliers(:,2);

    [A, B, bestTestLine, outliers, inliers] = RANSAC(x,y,0.1,20);
%     p = polyfit([A(1) B(1)], [A(2) B(2)], 1);
%     testline = polyval(p, x);
%     plot(x, testline, 'r-')


    lengths = vecnorm(inliers' - [10; 10]);
    [~, ia] = min(lengths);
    [~, ib] = max(lengths);
    end1 = inliers(ia,:);
    end2 = inliers(ib,:);
    line = [end1; end2];
    plot(line(:,1), line(:,2), 'g', 'LineWidth', 5)

end
```

```
toc

ylim([-1 3])
axis equal




legend('Dataset', 'RANSAC Fit Line')




function [A, B, bestTestLine, outliers, inliers] = RANSAC(x,y,d,n)
points = [x y];

bestInliersSoFar = 0;
bestTestLineSoFar = [];
bestInlierPointsSoFar = [];

% For each point pair
for i = 1:n
    % Randomly select the points
    A = points(randi([1, length(points)]),:);
    B = points(randi([1, length(points)]),:);

    % Find and plot the line between them
    p = polyfit([A(1) B(1)], [A(2) B(2)], 1);
    testline = polyval(p, x);

    That = [B(1) - A(1);B(2) - A(2);0]./vecnorm([B(1) - A(1);B(2) -
 A(2);0]);
    Khat = [0;0;1];
    Nhat = cross(That,Khat);

    inliers = 0;
    outliers = [];
    inlierPoints = [];
    % Test every point with this fit line
    for j = 1:length(points)
        r = horzcat(points(j,:) - A, 0);
        dist = dot(r,Nhat);
        if abs(dist) <= d
            inliers = inliers + 1;
            inlierPoints = vertcat(inlierPoints, points(j,:));
        else
            outliers = vertcat(outliers, points(j,:));
        end
```

```
    end

    if inliers > bestInliersSoFar
        bestPoint1SoFar = A;
        bestPoint2SoFar = B;
        bestTestLineSoFar = testline;
        bestInliersSoFar = inliers;
        bestOutliersSoFar = outliers;
        bestInlierPointsSoFar = inlierPoints;
    end
end
inliers = rmoutliers(bestInlierPointsSoFar);
endA = 0;
endB = 0;
bestTestLine = bestTestLineSoFar;
A = bestPoint1SoFar;
B = bestPoint2SoFar;
outliers = bestOutliersSoFar;
end
```

*Warning: Polynomial is badly conditioned. Add points with distinct X values, reduce the degree of the polynomial, or try centering and scaling as described in HELP POLYFIT.*
*Elapsed time is 1.099642 seconds.*



*Published with MATLAB® R2019a*