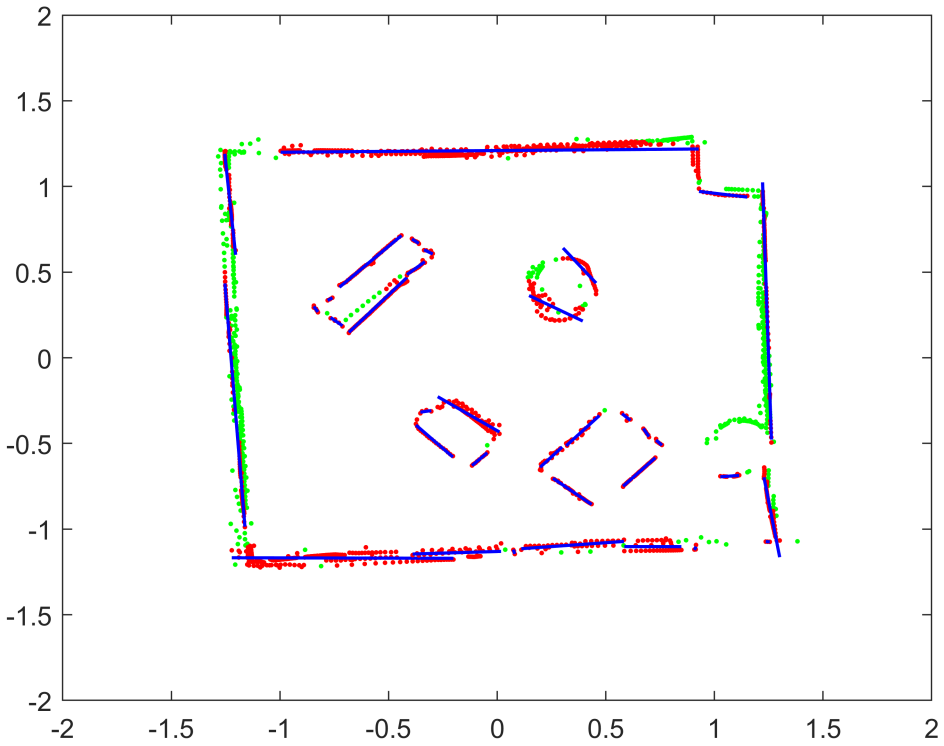
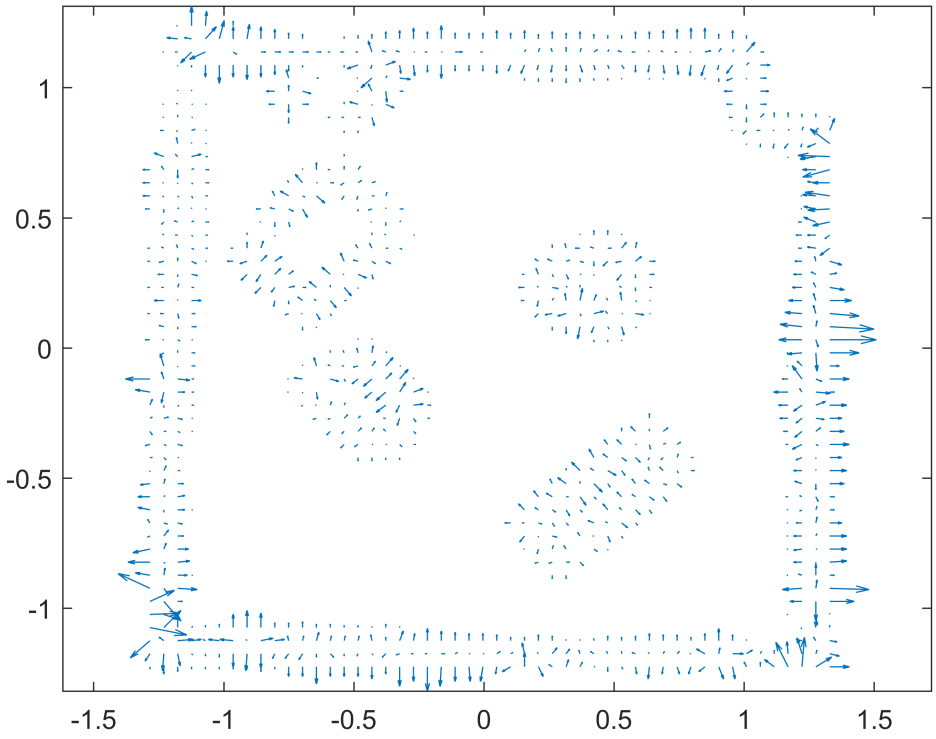
```
load("gauntlet.mat")
points = allpoints';
```

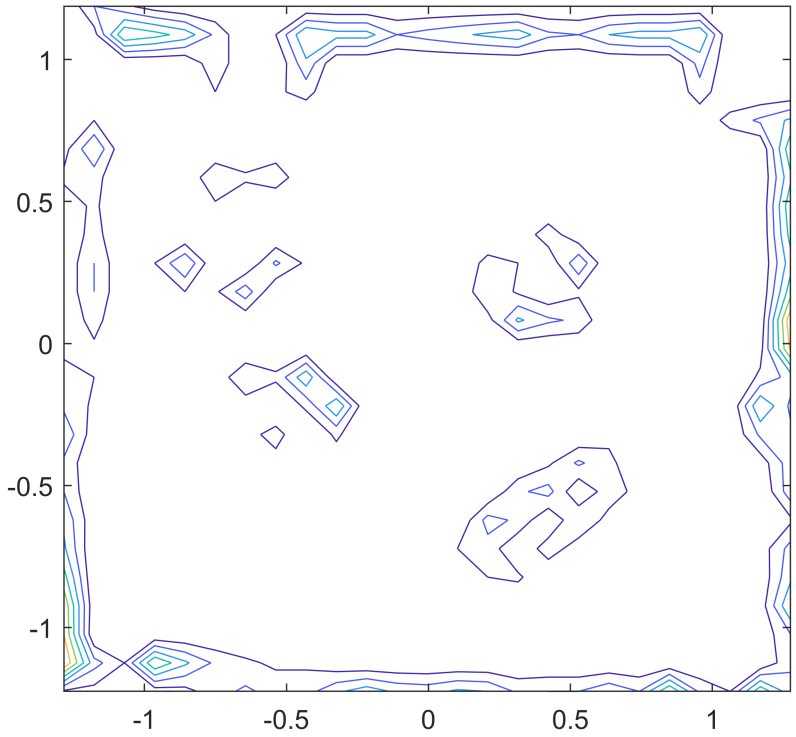
```
%points = [scan(:,1).*cos(scan(:,2)), scan(:,1).*sin(scan(:,2))];

outliersPlot = plot(points(:,1),points(:,2),'.',"Color","green","DisplayName","Outliers");
hold on

sampleSize = 2; % number of points to sample per trial
maxDistance = 0.01;% max allowable distance for inliers

fitLineFcn = @(points) polyfit(points(:,1),points(:,2),1); % fit function using polyfit
evalLineFcn = @(model, points) sum((points(:, 2) - polyval(model, points(:,1))).^2,2); % distar

clusters = clusterdata(points,'Criterion','distance',"linkage","single","cutoff",0.033);

for i = 1:max(clusters)
    cluster_i = points(clusters == i, :);

    if length(cluster_i) > 2
        [modelRANSAC, inlierIdx] = ransac(cluster_i,fitLineFcn,evalLineFcn,sampleSize,maxDistar

        modelInliers = polyfit(cluster_i(inlierIdx,1),cluster_i(inlierIdx,2),1);

        inlierPts = cluster_i(inlierIdx,:);
        x = [min(inlierPts(:,1)) max(inlierPts(:,1))];
        y = modelInliers(1)*x + modelInliers(2);
        inlierPlot = plot(inlierPts(:,1),inlierPts(:,2),'.',"Color","red", "DisplayName","Inlie
        bestFitPlot = plot(x, y, 'g-',"Color","blue","DisplayName","Best Fit Lines","LineWidth"
    end
end
hold off
xlim([-2 2])
ylim([-2 2])
```




```
% Load your x and y points into the following variables
X = points(:, 1);
Y = points(:, 2);

% Set the bin edges
numBins = 50;
xEdges = linspace(min(X), max(X), numBins+1);
yEdges = linspace(min(Y), max(Y), numBins+1);

% Compute the 2D histogram
[N,~,~] = histcounts2(X,Y,xEdges,yEdges);

% Compute the gradient
[FX,FY] = gradient(-N);

quiver(xEdges(1:end-1),yEdges(1:end-1),FX,FY, "AutoScaleFactor",2.5);
axis equal;
```

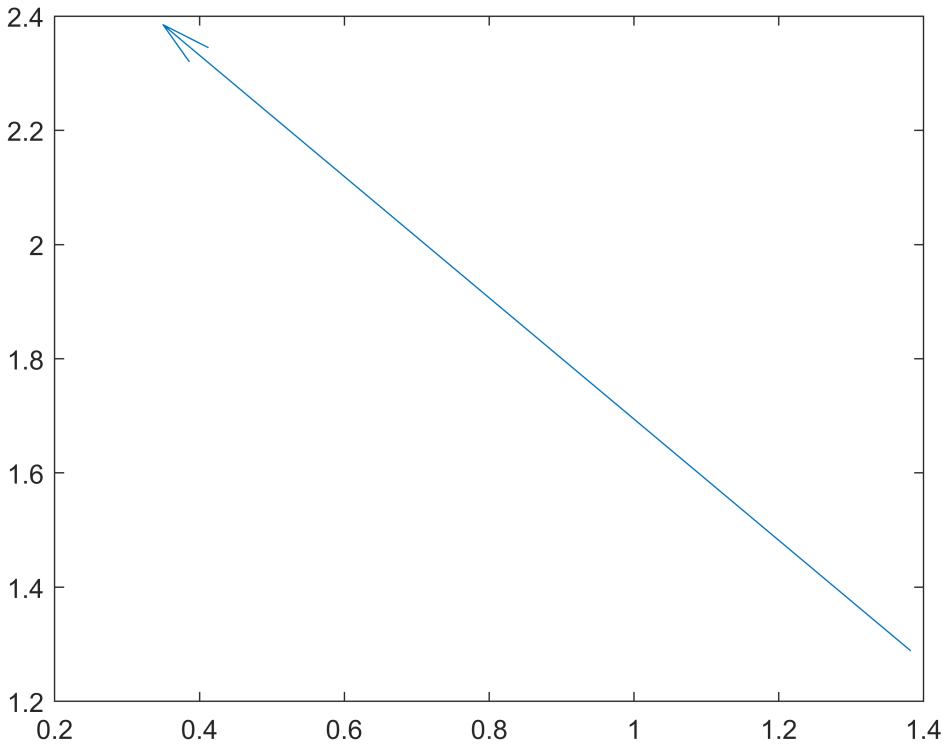



```
% Set the bin edges
numBins = 25;
xEdges = linspace(min(X), max(X), numBins+1);
yEdges = linspace(min(Y), max(Y), numBins+1);

% Compute the 2D histogram
[N,~,~] = histcounts2(X,Y,xEdges,yEdges);

% Compute the gradient
[Gx,Gy] = gradient(N);

% Generate the contour plot
contour(xEdges(1:end-1),yEdges(1:end-1),N);
axis equal;
```

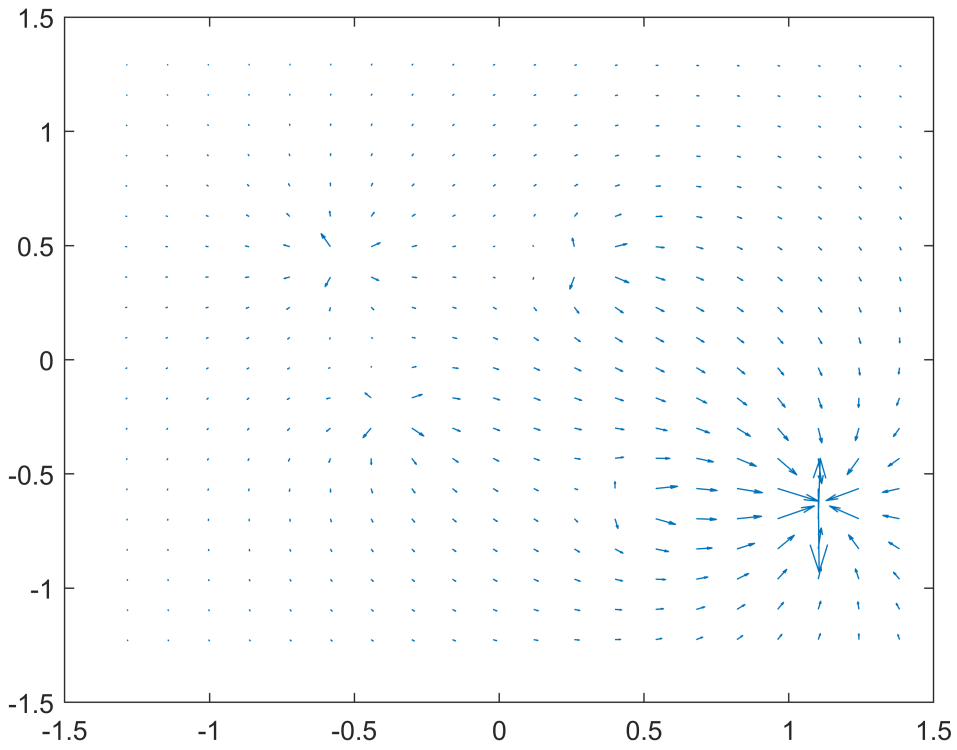



```
% Calculate the gradient of the points matrix
[Gx, Gy] = gradient(points);

% Create a grid of points
n_points = 20;
x = linspace(min(points(:,1)), max(points(:,1)), size(Gx,2));
y = linspace(min(points(:,2)), max(points(:,2)), size(Gx,1));
[X,Y] = meshgrid(x, y);

% Ensure that the input matrices have the same dimensions as the gradient matrices
X = X(1:size(Gx,1), 1:size(Gx,2));
Y = Y(1:size(Gy,1), 1:size(Gy,2));

% Create a function to represent the gradient field
grad_field = @(x,y) cat(2, interp2(X, Y, Gx, x, y), interp2(X, Y, Gy, x, y));

% Define a grid of points to evaluate the gradient field at
x = linspace(min(points(:,1)), max(points(:,1)), n_points);
y = linspace(min(points(:,2)), max(points(:,2)), n_points);
[X,Y] = meshgrid(x,y);

% Initialize arrays to store the x and y components of the gradient field
Gx = zeros(size(X));
Gy = zeros(size(Y));
```

```
% Loop through each point in the meshgrid and calculate the gradient field
for i = 1:numel(X)
    % Evaluate the gradient field function at the current point
    g = find_gradient(X(i), Y(i));

    % Store the x and y components of the gradient at the current point
    Gx(i) = g(1);
    Gy(i) = g(2);
end
```



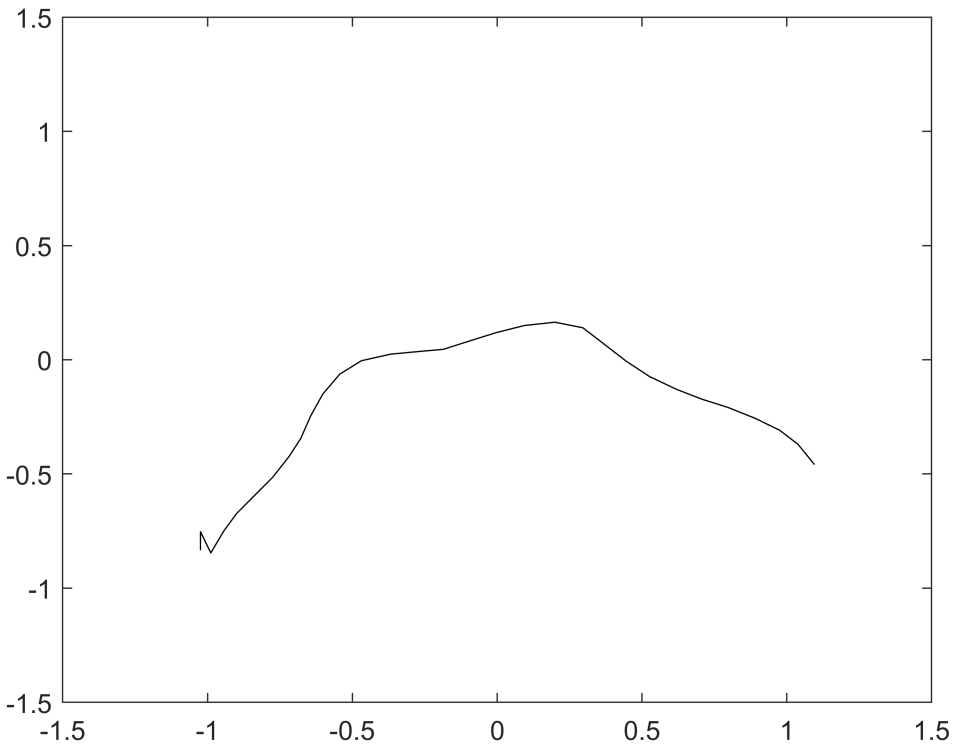

```
% Plot the quiver plot of the gradient field
quiver(X, Y, Gx, Gy, 2);
```




```
Z = [Gx ; Gy];
```

```
load("gauntle_encoder_data.mat");

d = 0.245;

%Measured Data
measured_t=encoder_data(:,1);
measured_l=encoder_data(:,2);
measured_r=encoder_data(:,3);

timestep_m=diff(measured_t);
dist_l_m=diff(measured_l);
dist_r_m=diff(measured_r);

velocity_left_measured = (dist_l_m./timestep_m);
velocity_right_measured = (dist_r_m ./ timestep_m);

LinVel = (velocity_right_measured+velocity_left_measured)/2;
AngVel = (velocity_right_measured-velocity_left_measured)/d;

x_position = zeros(1,length(measured_t));
y_position = zeros(1,length(measured_t));
theta = zeros(1,length(measured_t));
```

```
x_position(1) = (0.3960*cos(2.65*((0.1*0)+1.4)));
y_position(1) = -(0.99*sin((0.1*0)+1.4));
theta(1)=0;

for i = 1:length(measured_t)-1
    v_now = LinVel(i);
    ang_now = AngVel(i);
    theta_now = theta(i);
    x_position(i+1)=x_position(i) + v_now*cos(theta_now)*timestep_m(i);
    y_position(i+1)=y_position(i) + v_now*sin(theta_now)*timestep_m(i);
    theta(i+1)= theta_now + ang_now*timestep_m(i);
end

that_x = cos(theta);
that_y = sin(theta);

r_theta = -90;
R = [cosd(r_theta) -sind(r_theta); sind(r_theta) cosd(r_theta)];
coords = [x_position; y_position];
coords = coords'*R;

valid_rows = coords(:, 2) <= 2;
coords = coords(valid_rows, :);

%Measured Neato Path Plot
m_curve = plot(coords(:,1)-2, coords(:,2)-0.5,"-","Color","black");
xlim([-1.5 1.5])
ylim([-1.5 1.5])
```


1.5
1
0.5
0
-0.5
-1
-1.5
-1.5
-1
-0.5
0
0.5
1
1.5