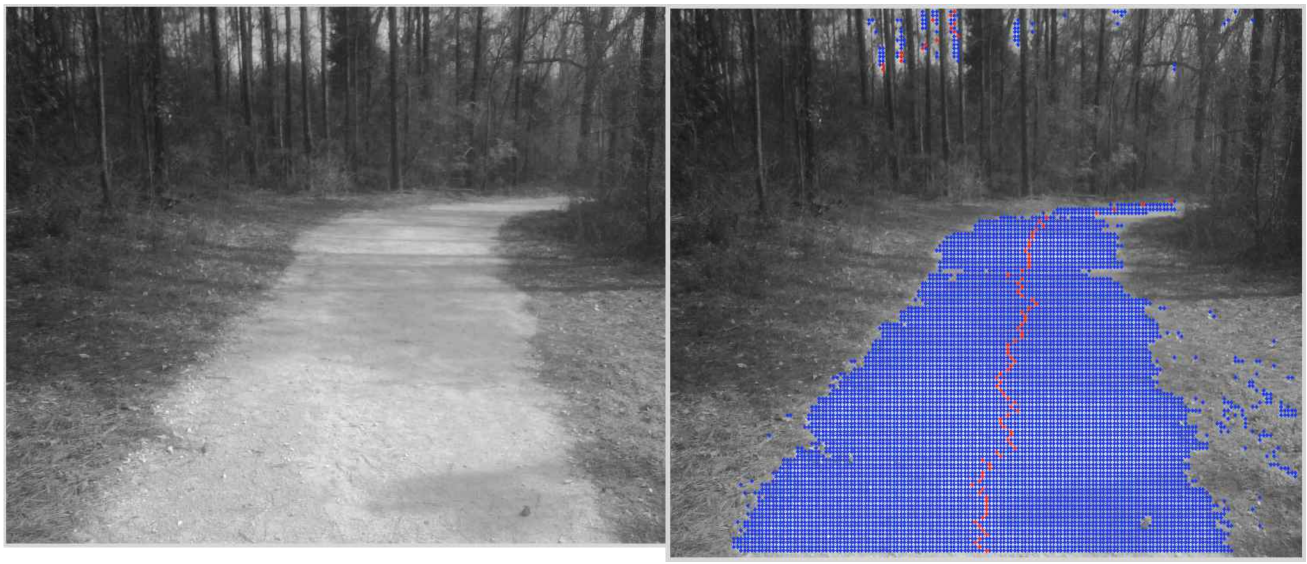
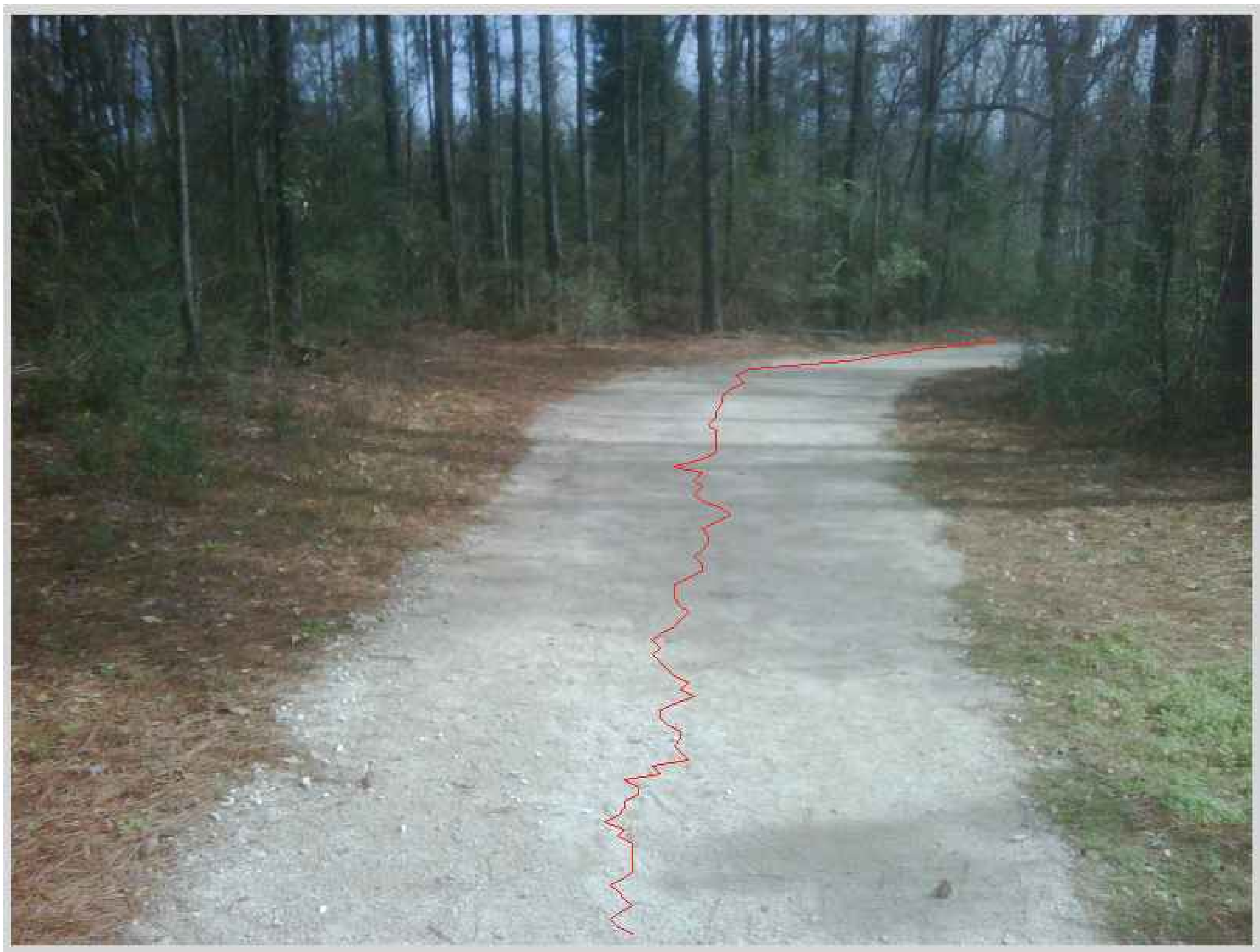
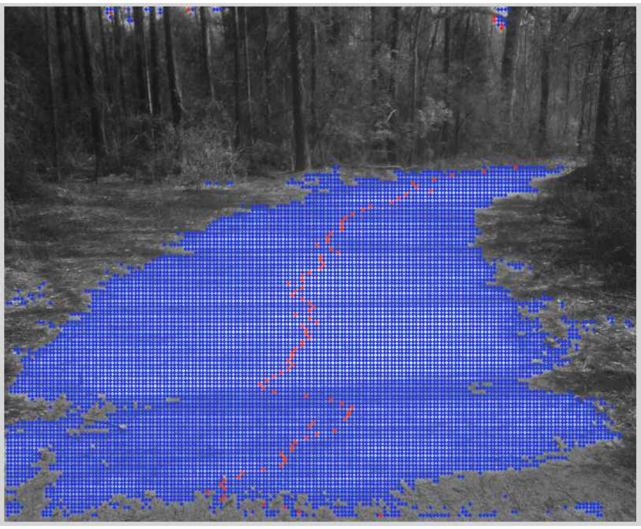
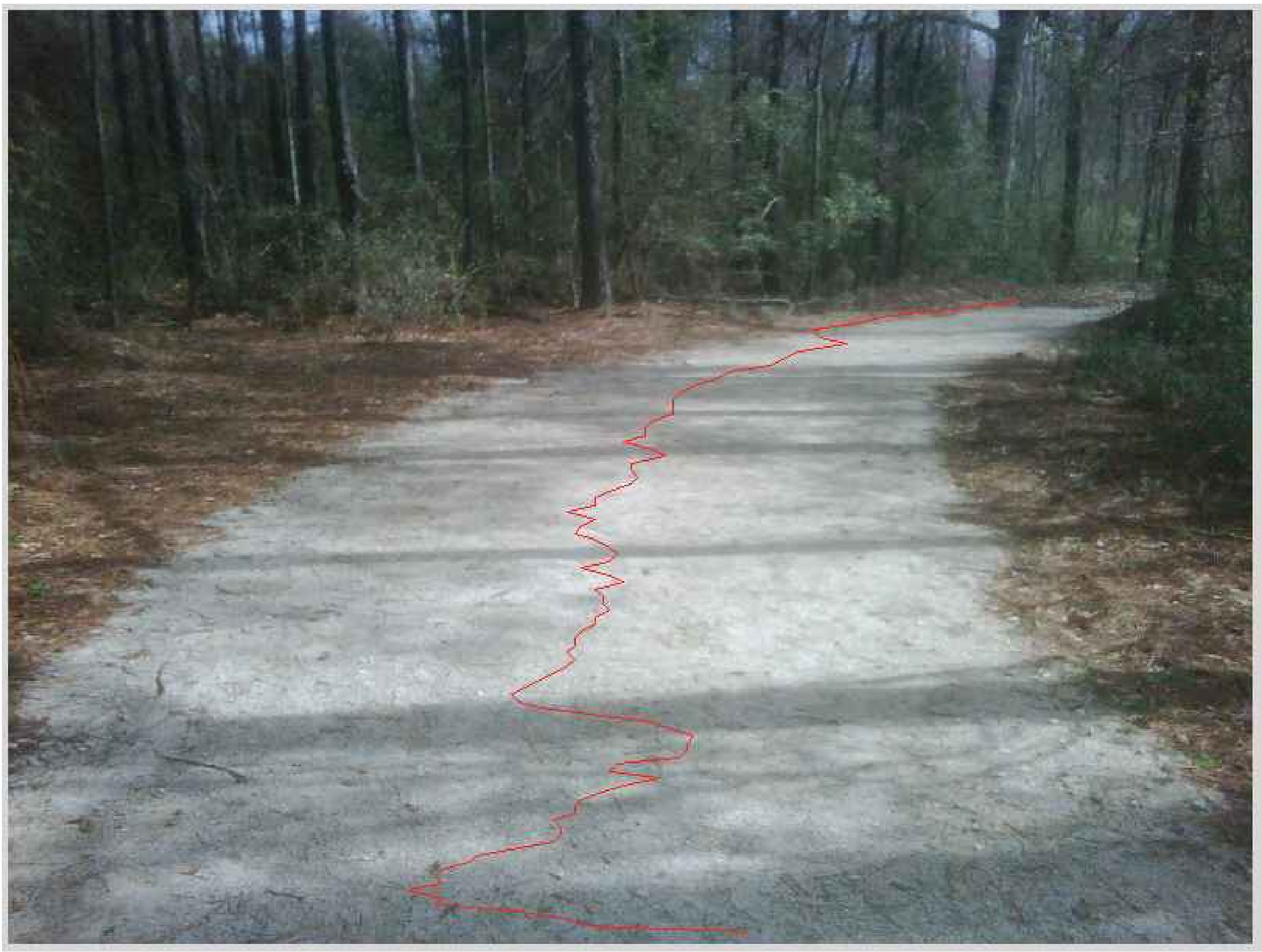
Eric Broshears
HW4
Autonomous Mobile Robots
Dr. Roppel

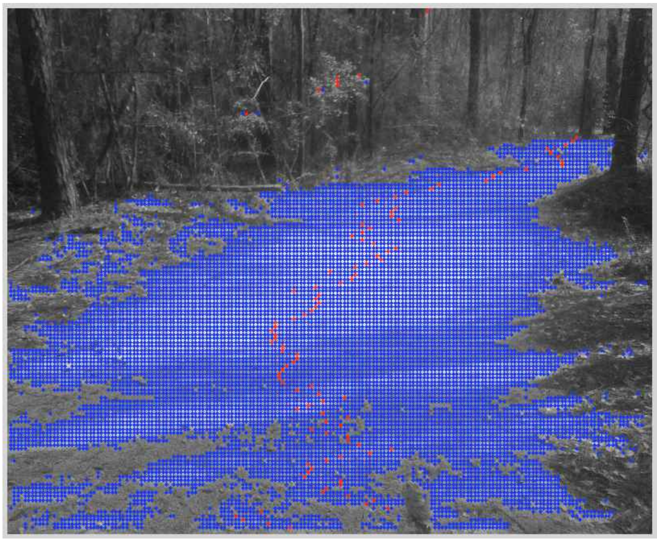
This homework has exposed many of the issues when it comes to visual navigation. Because the edges of the path were not clearly defined, a hough transform would've been difficult to do. The algorithm used made a block size (~3x3 pixel block) and went row by row of the picture and calculated the average pixel intensity. A threshold was set, which was made according to the average intensity of the lower half and middle third of the picture, which was assumed to be comprised mostly of the path. Then, all the pixels that were above this threshold were plotted in blue with the median of each line plotted in red. Connected the red dots gave a relatively accurate path to follow. There were several issues when there were bright spots off the path, such as the sun reflecting off of leaves or the water. Since the vanishing point was usually around the top third of the image, the path was assumed to be ended here, which is an acceptable assumption since the next image will take place before the path ends on the current image, anyways. Future improvements that could be made is to smooth the line to eliminate all the jaggedness as well as to eliminate the random outliers caused by the shadows on the path and the bright spots off the path. Also, coupled with an IMU, the vanishing point and the width of the path could be predicated more accurately.

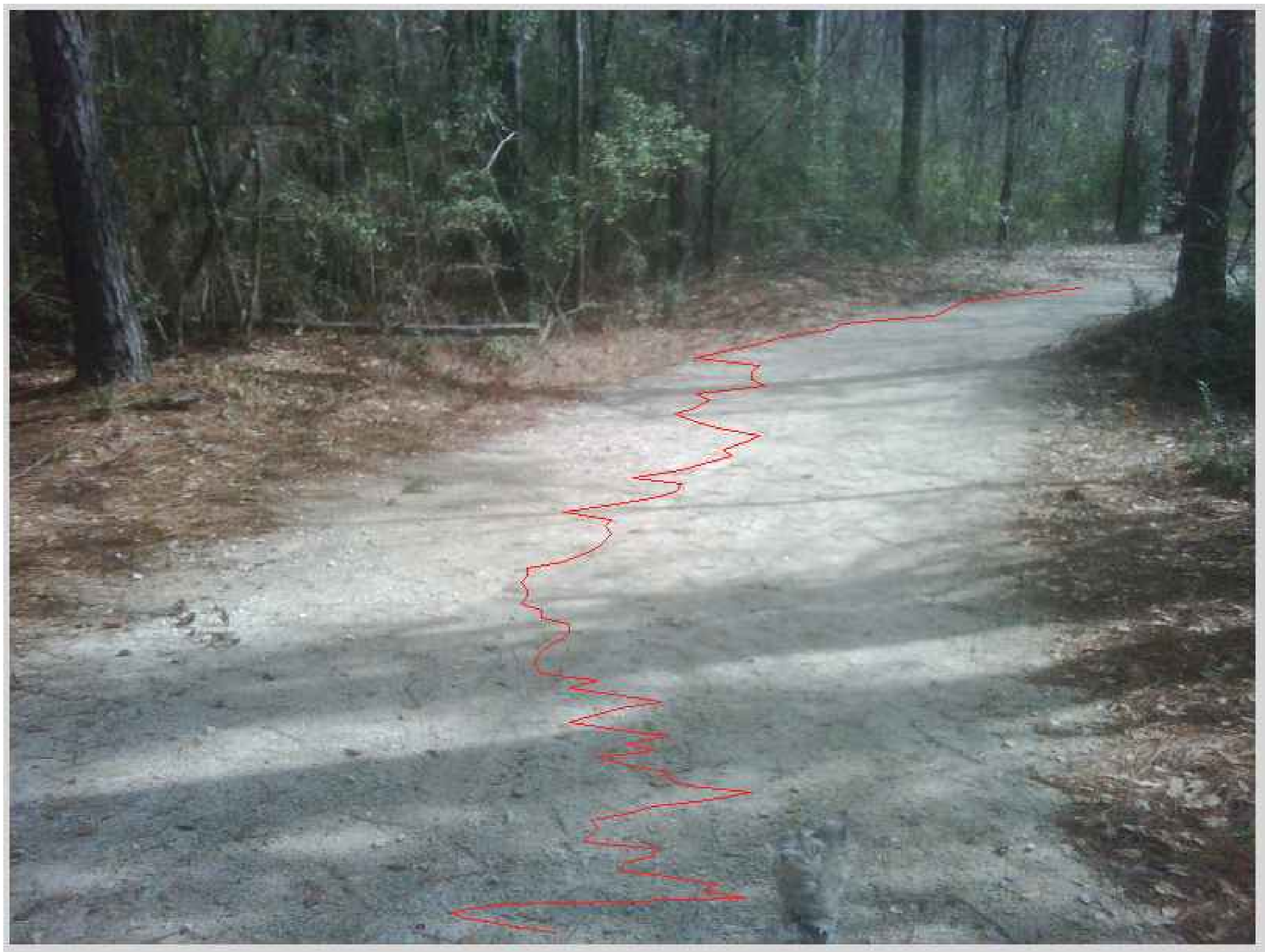


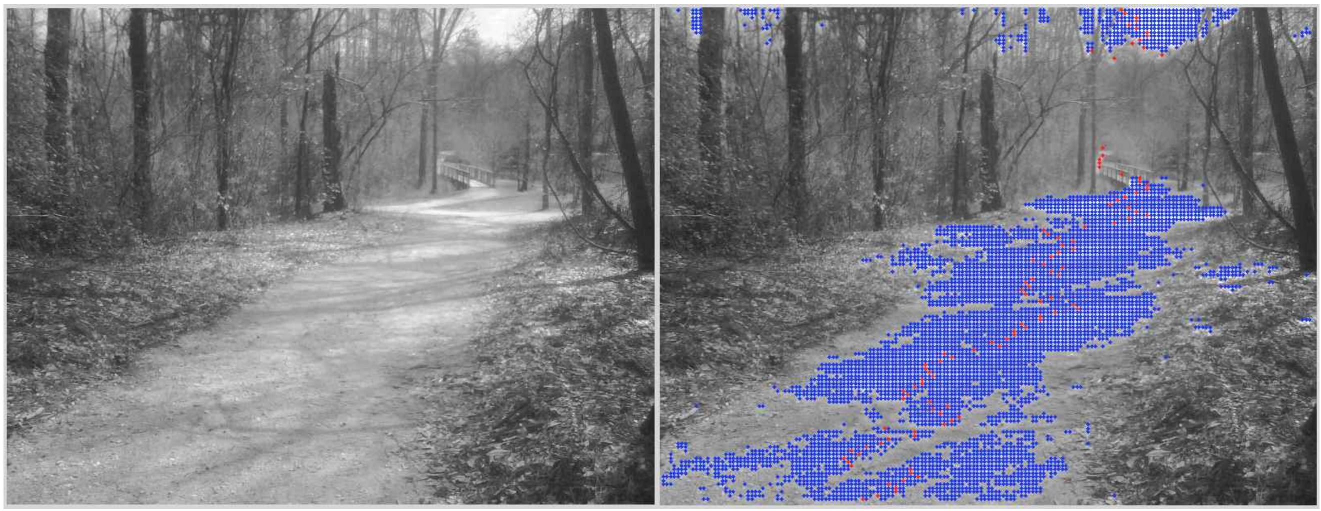


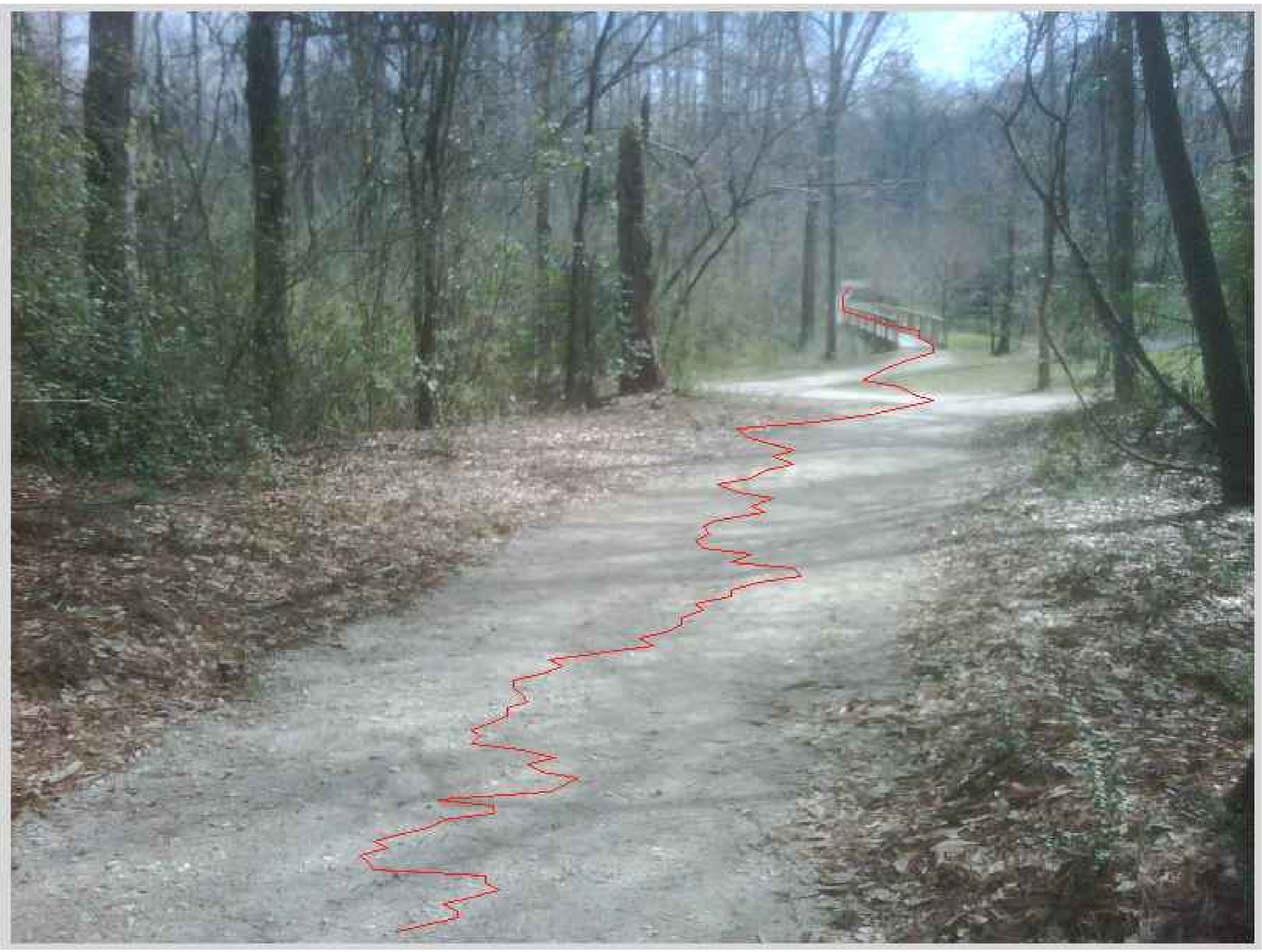


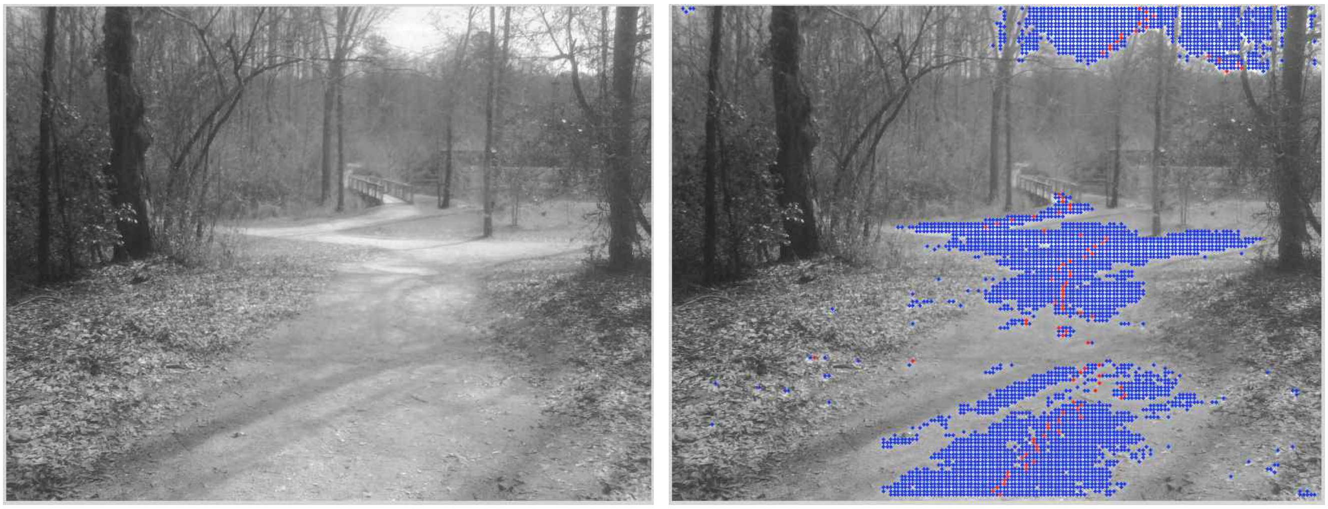


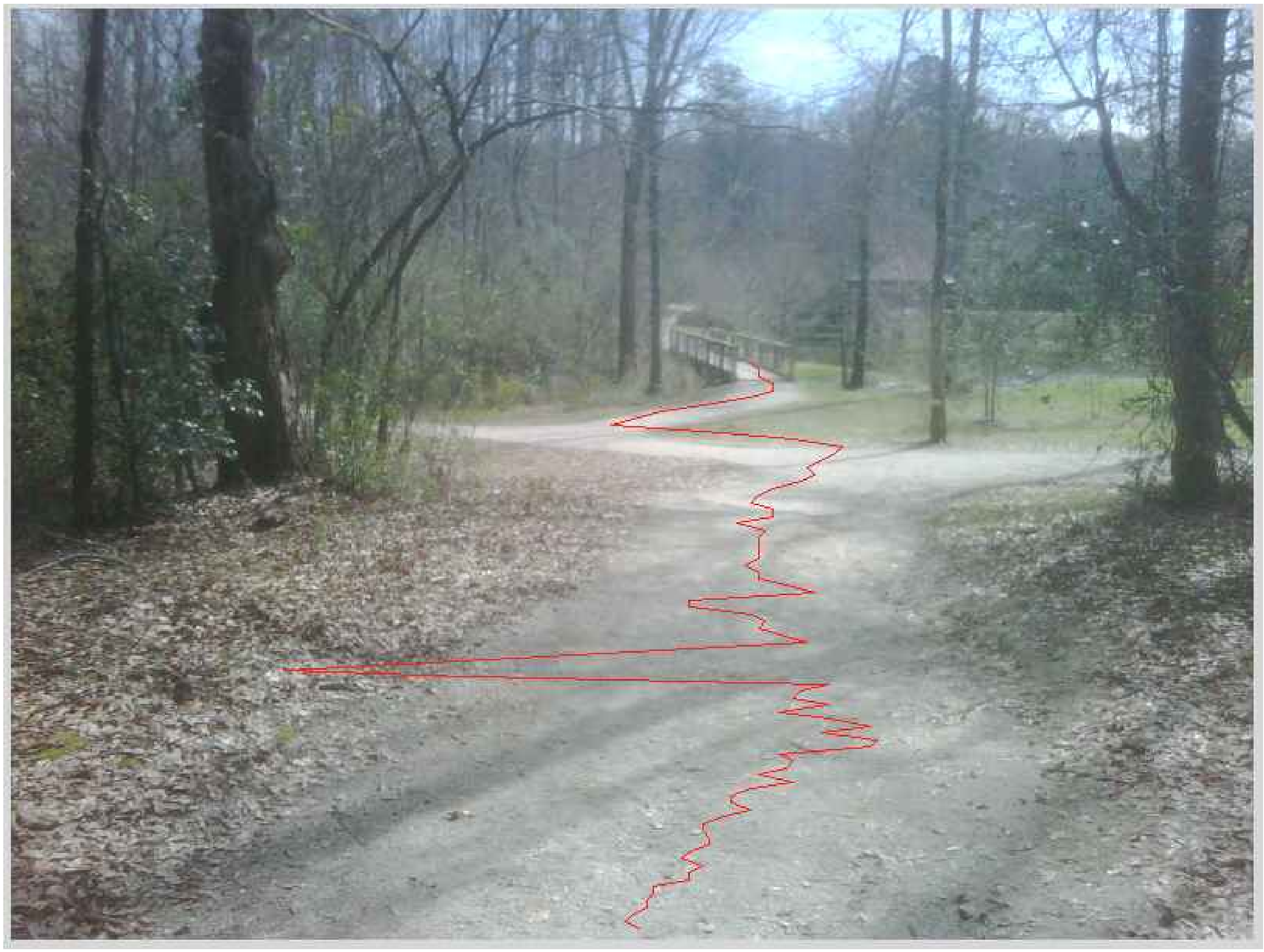


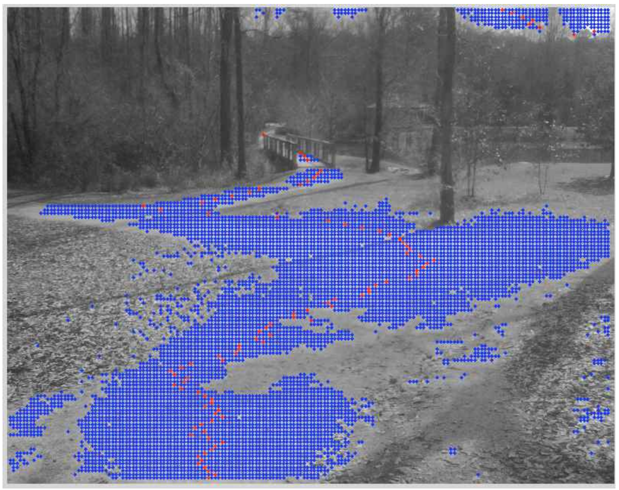


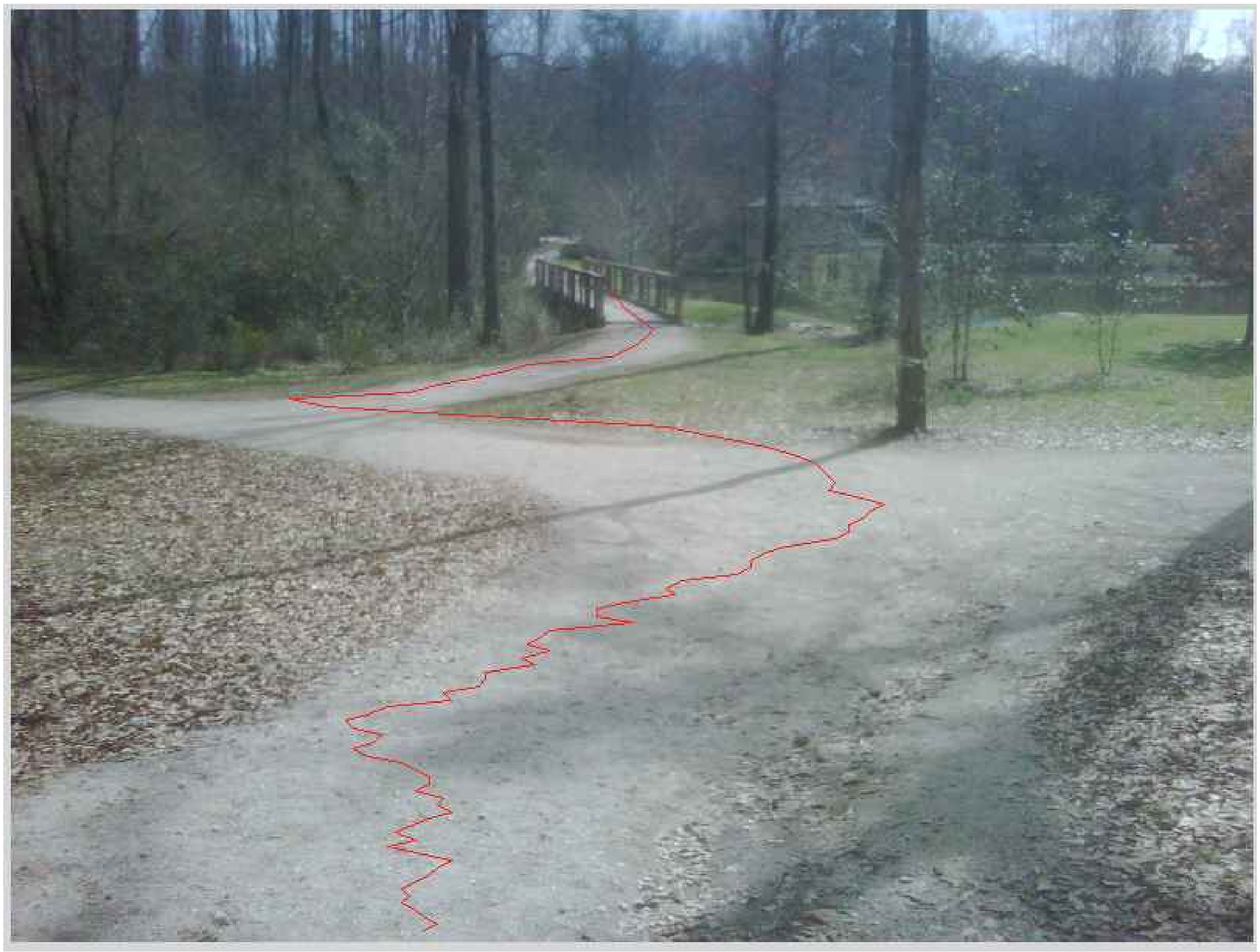


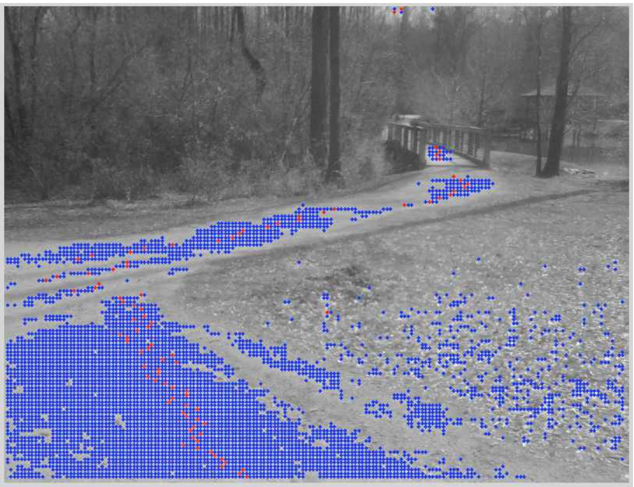


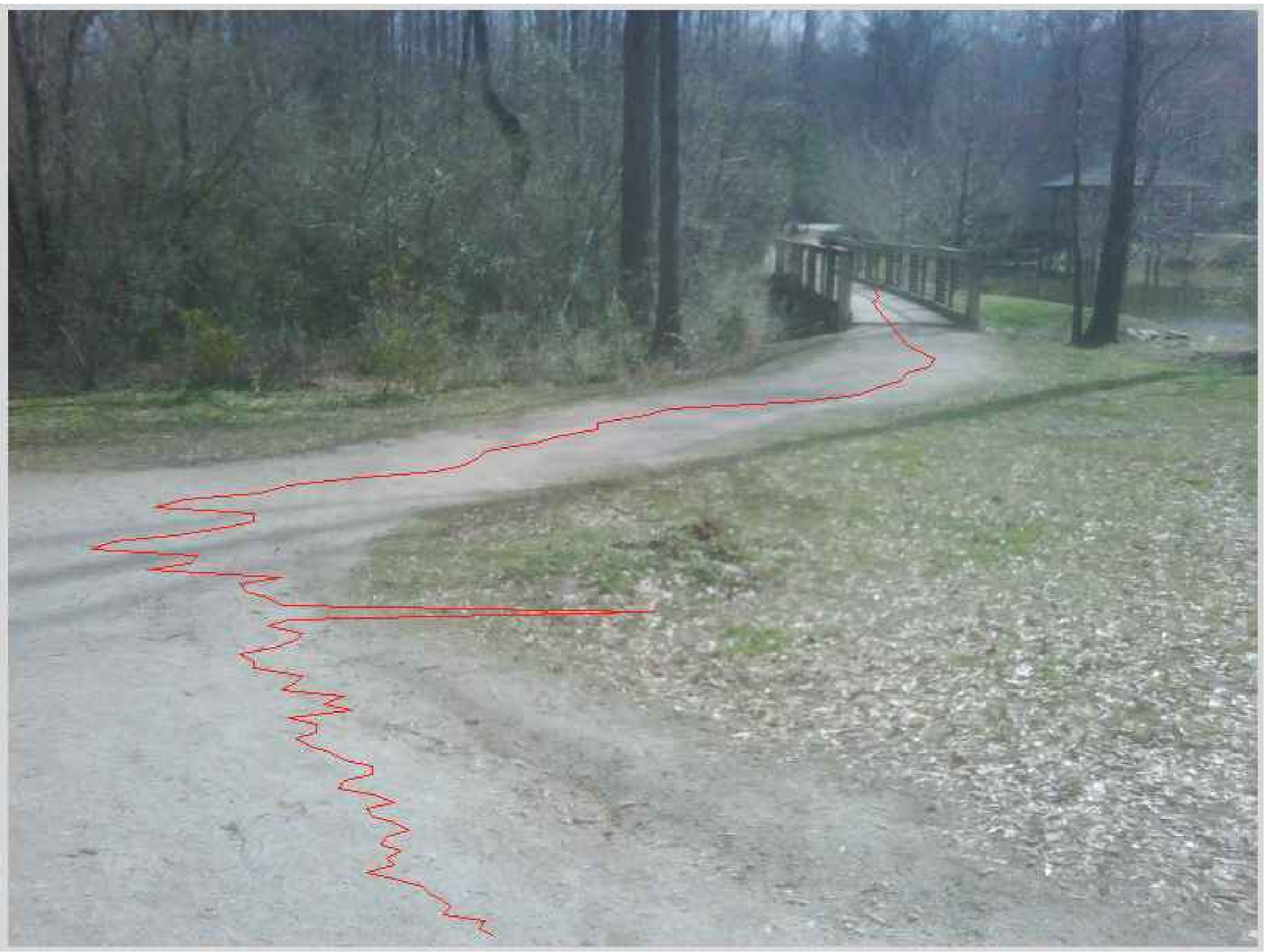


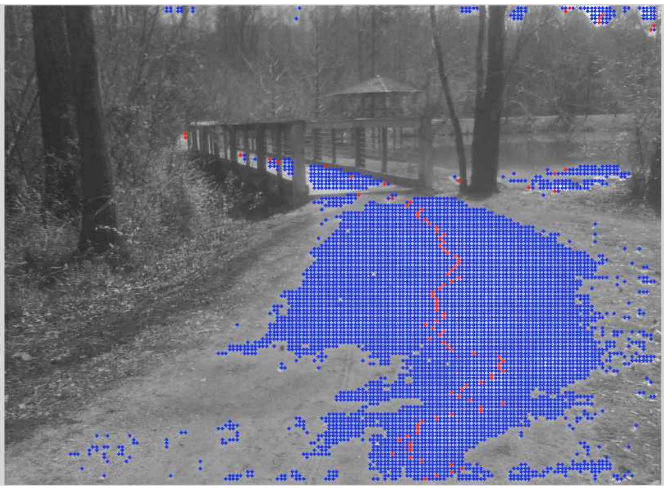


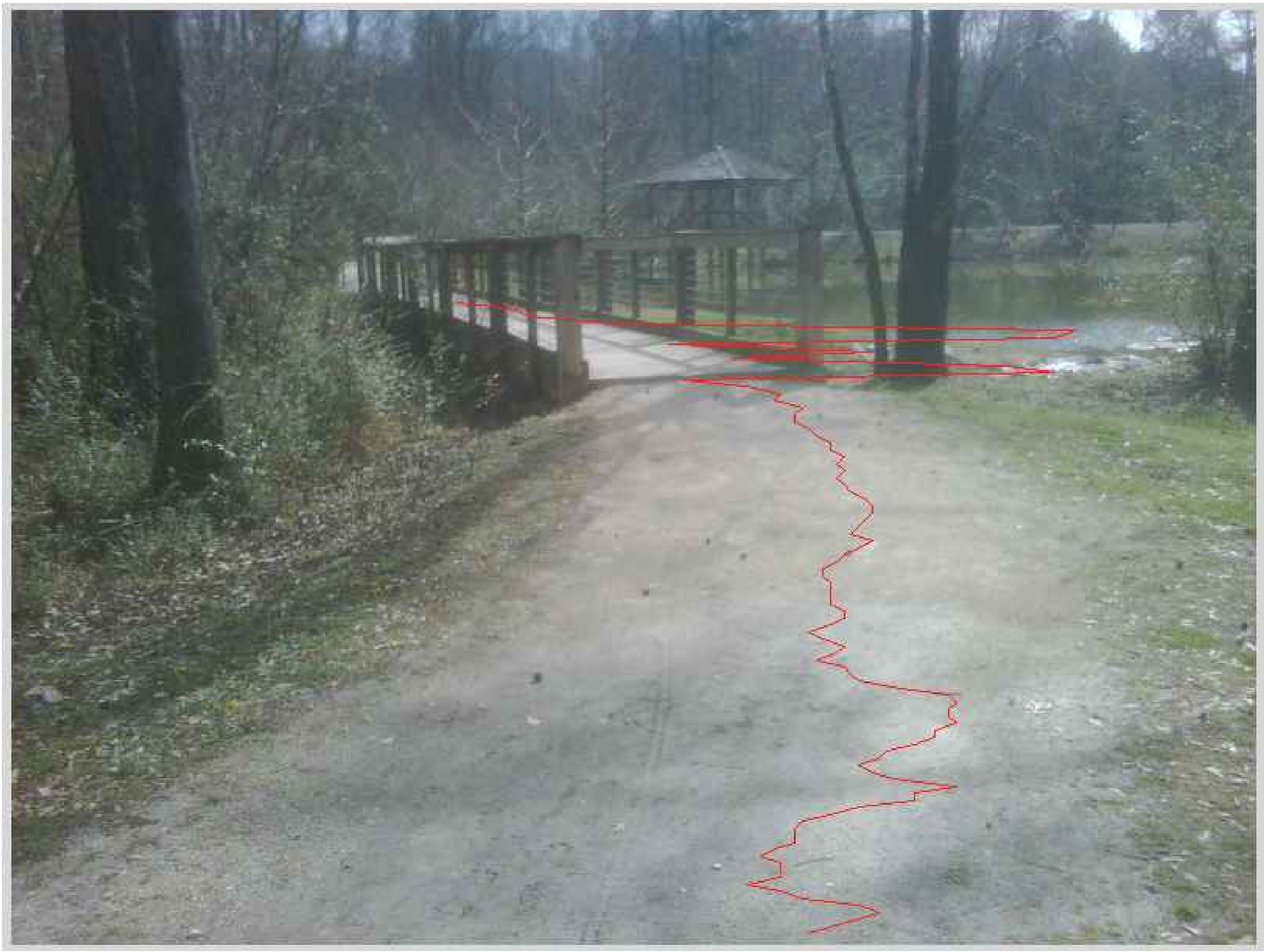


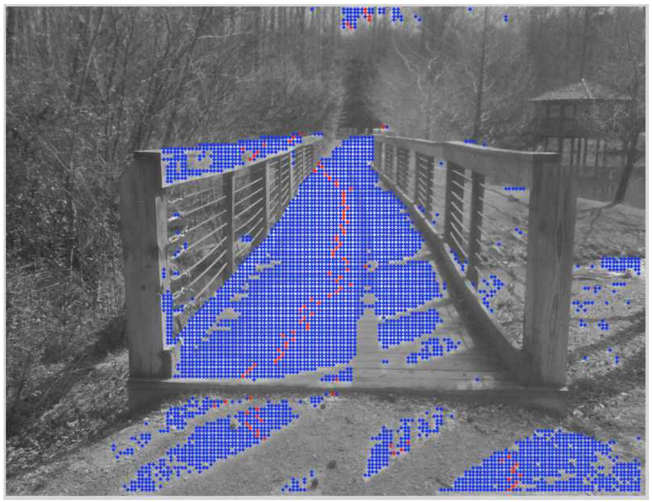


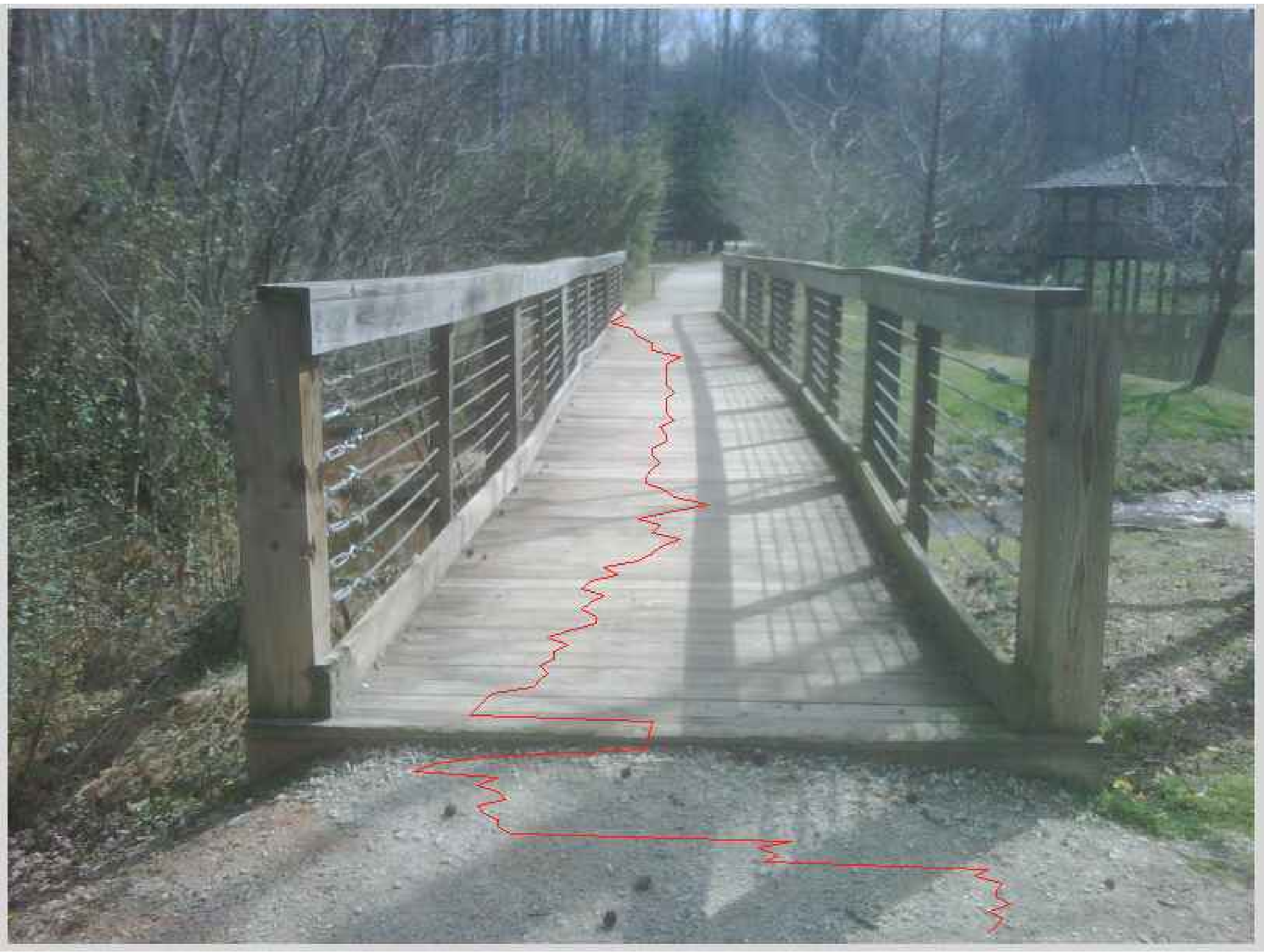












```matlab
% PathDetection2.m


close all
clear
clc

addpath /Users/ericbroshears/Desktop/Classes/Autonomous' Mobile Robots'/Homework/HW4/hw4photos


% I = imread('p1.jpg'); tol = 130; topLimit = 16; % pathAvg = 180
% I = imread('p2.jpg'); tol = 100; topLimit = 7; % pathAvg = 150
% I = imread('p3.jpg'); tol = 100; topLimit = 15; % pathAvg = 125
% I = imread('p4.jpg'); tol = 150; topLimit = 16; % pathAvg = 150
% I = imread('p5.jpg'); tol = 160; topLimit = 19; % pathAvg = 160
% I = imread('p6.jpg'); tol = 140; topLimit = 10; % pathAvg = 140
% I = imread('p7.jpg'); tol = 160; topLimit = 3; % pathAvg = 150
% I = imread('p8.jpg'); tol = 140; topLimit = 10; % pathAvg = 145
I = imread('p9.jpg'); tol = 140; topLimit = 16; % pathAvg = 140

G = rgb2gray(I);
Gt = G';

testArea = Gt(600:1100,700:1200);

pathAvg = mean(mean(testArea));


blockSize = 9;
border = ceil(blockSize/2);

[imageY,imageX] = size(G);

slideX = floor((imageX-(2*border))/blockSize);
slideY = floor((imageY-(2*border))/blockSize);


jj = 0;

for kk = 1:slideY

    j = 0;

    for k = 1:slideX

        centerPixel = [border + j*blockSize, imageY - border - jj*blockSize];

        block = Gt(centerPixel(1) - (border-1) : centerPixel(1) + (border-1), ...
            centerPixel(2) - (border-1) : centerPixel(2) + (border-1));

        intensity(k,kk) = mean(mean(block));

        j = j+1;

    end

    jj = jj+1;

end



figure
imshow(G)


figure
imshow(G)
hold on

mCount = 1;

for k = 1:slideY

    intRow = k; % max(intRow) = slideY
    imageRow = intRow*blockSize + border;

    avgInt = mean(intensity(:,intRow));

    count = 0;
    point = [];

    for kk = 1:slideX
        if intensity(kk,intRow) > tol
```

```
            plot((kk*blockSize),(imageY-imageRow),'b.')

            count = count+1;

            point(count,:) = [(kk*blockSize) (imageY-imageRow)];

        end
    end

    if ~isempty(point)
        midPoint = ceil(count/2);
        plot(point(midPoint,1),point(midPoint,2),'r.')
        midPoints(mCount,:) = point(midPoint,:);
        mCount = mCount+1;
    end

end



% close all
figure
imshow(I)
hold on
for k = 1:length(midPoints)-topLimit
    plot([midPoints(k,1) midPoints(k+1,1)],[midPoints(k,2) midPoints(k+1,2)],'r-')
end
```