Table of Contents

.... 1
Initialization of MAP Matrix with Boundary Points, Obstacles, and Waypoints 1
A* function 2
Visualization 3
Reduce Redudant WPs 3
Address Changing Altitudes 3
Format Path for Waypoint file 3
Servo 4
Print Waypoints into waypoint file format 4
End Time 4

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% A* ALGORITHM Demo
% Interactive A* search demo
% 04-26-2005

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 4/18/2020
% New: added do servo command
% Current:
% Test is for a 1417x1143 matrix
% runs A* in the bounded area with obstacles
% inaccuracies of +/- 1.1m
% combined all parsing data into a function that only needs the file
 name
% extracts optimal path using parallel computation
% Reduce optimal path with reduced jaggity lines
% Outputs: computation time, a plot of the total path, reduced optimal
% path, and waypoint file for Mission Planner
% Added UGV Drop pos
% Results: Able to output waypoint file in the correct format
% Next Step:
% 1. Testing in mission planner
% 2. Could reduce more WPs
```

Initialization of MAP Matrix with Boundary Points, Obstacles, and Waypoints

```
close all; clc;

tStart = tic; % start clock1 to calculate runtime

%Get json data into matlab
fname = 'test.json';

%Given File name
```

```
%Return location of each waypoint, MAP Matrix with obstacles and
 boundary lines added, and waypoint altitudes
[MAP,num_WP,x_WP,y_WP,n,m,alt_WP] = parse_all_data(fname);
```

A* function

```
%Given the MAP Matrix and location of first and second point
%Return optimal path between those two points in the MAP Matrix


%Initialize
Optimal_path = cell(1,num_WP(1)-1);
Timer = zeros(1,num_WP(1)-1);

%Initialize Waypoint Variables
%Starting point
x_WP1 = zeros(1,num_WP(1)-1);
y_WP1 = zeros(1,num_WP(1)-1);
%End point
x_WP2 = zeros(1,num_WP(1)-1);
y_WP2 = zeros(1,num_WP(1)-1);
for i = 1:num_WP(1)-1
x_WP1(i) = x_WP(i);
y_WP1(i) = y_WP(i);
x_WP2(i) = x_WP(i+1);
y_WP2(i) = y_WP(i+1);
end

%A Star
parfor i = 1:(num_WP(1)-1)
tic;    % start clock2 to calculate runtime
Optimal_path{i} =
 A_Star_function(MAP,x_WP1(i),y_WP1(i),x_WP2(i),y_WP2(i),n,m);
Timer(i) = toc; % end clock2 to calculate runtime
fprintf('Elapsed time from waypoint %d to %d is %8.6f seconds\n',i,i
+1,Timer(i));
end
```

```
Starting parallel pool (parpool) using the 'local' profile ...
Connected to the parallel pool (number of workers: 8).
Elapsed time from waypoint 4 to 5 is 8.145659 seconds
Elapsed time from waypoint 10 to 11 is 10.646973 seconds
Elapsed time from waypoint 6 to 7 is 21.083783 seconds
Elapsed time from waypoint 14 to 15 is 21.780380 seconds
Elapsed time from waypoint 13 to 14 is 7.246855 seconds
Elapsed time from waypoint 2 to 3 is 36.694317 seconds
Elapsed time from waypoint 3 to 4 is 39.235781 seconds
Elapsed time from waypoint 9 to 10 is 62.341370 seconds
Elapsed time from waypoint 8 to 9 is 115.010813 seconds
Elapsed time from waypoint 7 to 8 is 33.996095 seconds
Elapsed time from waypoint 1 to 2 is 169.023718 seconds
Elapsed time from waypoint 12 to 13 is 216.629663 seconds
Elapsed time from waypoint 5 to 6 is 198.519083 seconds
Elapsed time from waypoint 11 to 12 is 24.401695 seconds
```

Visualization

%Plot the Optimal Path! figure(1) axis([1 n 1 m]) grid on; hold on; ylabel('Latitude') xlabel('Longitude')

%Plots visual dots for line boundary and obstacles for i = 1:n for j = 1:m if MAP(i,j) == -1 plot(i+0.5,j+0.5,'r.'); %plots obstacles and boundary lines end end end

%Visualization of the Optimal Path for i = 1:num_WP(1)-1 j=size(Optimal_path{1,i},1); %Plots starting point plot(Optimal_path{1,i}(j,1)+.5,Optimal_path{1,i}(j,2)+.5,'r.'); %Plots end point plot(Optimal_path{1,i}(1,1)+.5,Optimal_path{1,i}(1,2)+.5,'r.'); plot(Optimal_path{1,i}(:,1)+.5,Optimal_path{1,i}(:,2)+.5); end

Reduce Redudant WPs

```
Reduced_path = cell(1,num_WP(1)-1);
for i = 1:num_WP(1)-1
Reduced_path{i} = reduce_collinear_points(Optimal_path{1,i}
(:,1)',Optimal_path{1,i}(:,2)');
end
```

Address Changing Altitudes

```
alt = cell(1,num_WP(1)-1);
for i = 1:num_WP(1)-1
alt{i} =
 assign_alt(x_WP1(i),y_WP1(i),x_WP2(i),y_WP2(i),alt_WP(i),alt_WP(i
+1),Reduced_path{1,i});
end
```

Format Path for Waypoint file

```
%Combine all Reduced waypoints and altitudes
Combined_Reduced = cat(1,Reduced_path{:});
total_num_WP = size(Combined_Reduced);

%Mission Planner reads waypoint files in meters
Combined_alt_ft = cat(1,alt{:});
%Convert ft to m
Combined_alt_m = Combined_alt_ft * 0.3048;


%Initialization
longitude = zeros(total_num_WP);
latitude = zeros(total_num_WP);

%Convert MAP Matrix cords back to GPS cords
for i = 1:total_num_WP(1)
    [longitude(i), latitude(i)] =
 MAP2GPS(Combined_Reduced(i,1),Combined_Reduced(i,2));
```

```
end
waypoints = zeros(length(Combined_Reduced),4);
%Orders waypoints into Waypoint file format
for i = 1:1:total_num_WP(1)
    waypoints(i,:) = [i,latitude(i), longitude(i), Combined_alt_m(i)];
end
```

Servo

```
servo_port_number = 3; % Change to the port that the drop box servoe
 is plugged into
servo_PWM = 1500; %Choose values between 1000 and 2000
servo = [total_num_WP(1)+1,servo_port_number,servo_PWM,0,0,0];
```

Print Waypoints into waypoint file format

```
fileID = fopen('waypoint.waypoints','w');
fprintf(fileID, 'QGC WPL 110\n');
fprintf(fileID, '0 1 0 16 0 0 0 0 38.143181 -76.430619 0.000000 1\n');
fprintf(fileID, '%d 0 3 22 0.00000000 0.00000000 0.00000000 0.00000000
 %8.5f %8.5f %f 1\n',waypoints(1,:)');
fprintf(fileID, '%d 0 3 16 0.00000000 0.00000000 0.00000000 0.00000000
 %8.5f %8.5f %f 1\n',waypoints(2:(total_num_WP(1)-1),:)');
fprintf(fileID, '%d 0 3 16 0.00000000 0.00000000 0.00000000 0.00000000
 %8.5f %8.5f %f 1\n',waypoints(total_num_WP(1),:)');
fprintf(fileID, '%d 0 3 183 %8.5f %8.5f 0.00000000 0.00000000 %8.5f
 %8.5f %f 1\n',servo(1,:)');
fclose(fileID);
```

End Time

```
tEnd = toc(tStart); % end clock1 to calculate runtime
fprintf('Total time is %8.6f seconds\n',tEnd);
```

```
Total time is 265.455757 seconds
```

Copyright 2009-2010 The MathWorks, Inc.
Published with MATLAB® R2020b