ME145 Robotic Planning and Kinematics: Lab 2

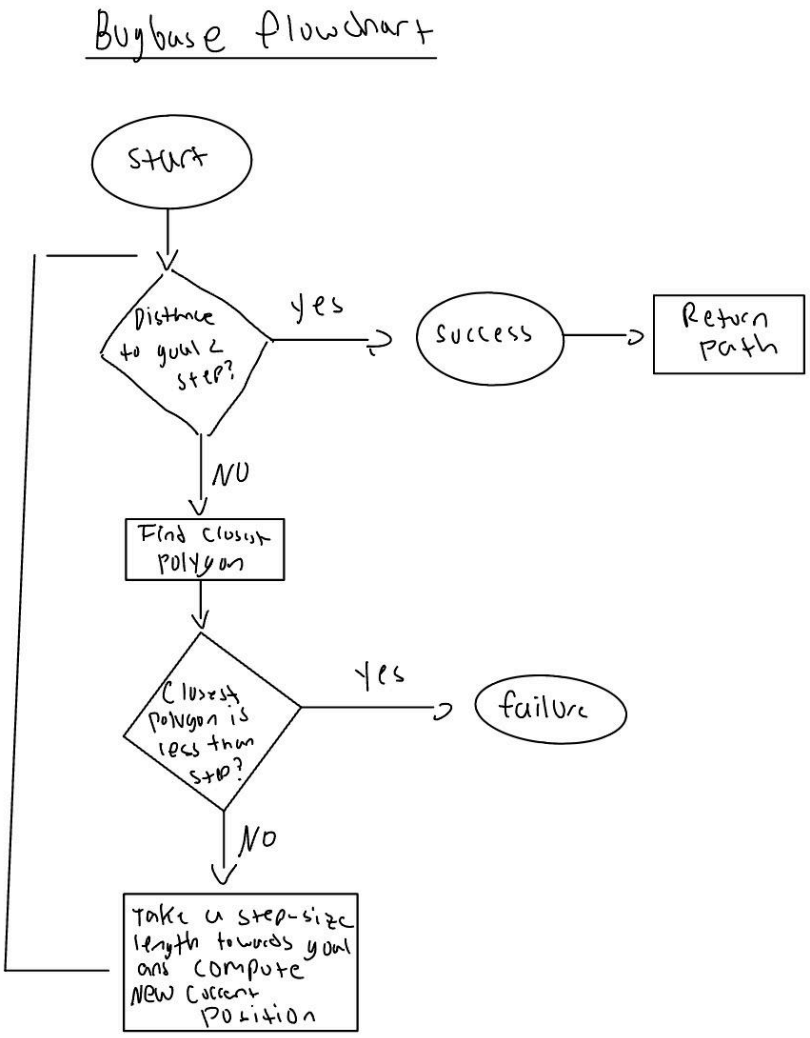
Bug 1 Algorithm

Eric Perez

Spring 2024

April 22, 2024

i) BugBase Flowchart



Implementing BugBase:

```
% Eric Perez
% ME 145 Lab 2

function [sequence] = bugBase(start,goal,sSize,obstaclesList)
robotPosition = start;
sequence = [start];
% Distance from start to goal
distanceToGoal = @(p1,p2) norm(p1 -p2);
%distanceToGoal = norm(robotPosition - goal)
sizeOb = size(obstaclesList);
numObstacles = sizeOb(2);
q = robotPosition;
obstacleCollision = 0;
 while distanceToGoal(robotPosition,goal) > sSize

    for i=1:numObstacles
        [distance,xvalue,yvalue] = computeDistancePointToPolygon(cell2mat(obstaclesList{i}'),q);
        distanceArray(i) = distance;
        xvalueArray(i) = xvalue;
        yvalueArray(i) = yvalue;
    end

closestObstacle = min(distanceArray);
indexclosestObstacle = find(distanceArray == closestObstacle);
closestObstacle = cell2mat(obstaclesList{indexclosestObstacle}');

    nextPosition = robotPosition + sSize*(goal-robotPosition) / distanceToGoal(robotPosition, goal);
    sequence = [sequence; nextPosition];
    robotPosition = nextPosition;
```

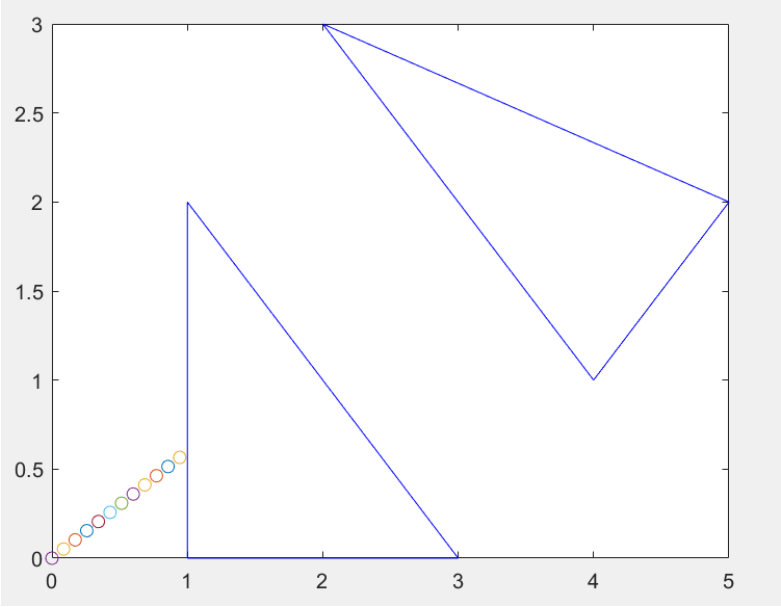
```
% Attempting to use tangent vector to polygon
q = robotPosition;
P = closestObstacle;
[u,minSeg,MinVertexDistance] = computeTangentVectorToPolygon(P,q)

      if minSeg < sSize || MinVertexDistance < sSize
           nextPosition = robotPosition + sSize * u;
           sequence = [sequence; nextPosition];
           robotPosition = nextPosition;
           obstacleCollision = 1;
           error('Failure: There is an obstacle lying between the start and goal');
      % break
      end


 end
  xvalueSequence = sequence(:,1);
  yvalueSequence = sequence(:,2);
  xvalueStart = start(1);
  yvalueStart = start(2);
  xvalueGoal = goal(1);
  yvalueGoal = goal(2);
  plot(xvalueStart,yvalueStart,'o');
  hold on
  plot(xvalueGoal,yvalueGoal,'o');
  plot(xvalueSequence,yvalueSequence,'o');
  disp('Success')

end
```

Example:

Input: [sequence] = bugBase([0 0],[5 3],[0.1], {{[1, 2], [1, 0], [3, 0]}, {[2, 3], [4, 1],[5, 2]}})

Output:

```
Error using bugBase (line 58)
Failure: There is an obstacle lying between the start and goal
```



ii) The BugBase algorithm detects if there is an obstacle in front of its path, and if there is then it fails. The Bug 1 algorithm will detect the obstacle in its path and once reached, it will circumnavigate the obstacle to find the shortest path towards the goal. To accomplish this, previous functions such as computeTangentVectorToPolygon will be used. This function outputs the minimum segment distance, minimum vertex distance, and the U vector. The U vector will be used to circumnavigate the obstacle as it will run parallel to segments and turn on vertices. Once the point has gone fully around the obstacle, it will then travel back to the point on the obstacle that offered the shortest path to the goal. The point will then shoot off and repeat this process if another obstacle is detected.

iii) Full Bug1 code:

```
%Eric Perez
%Lab 2: Bug 1

function [sequence,pathLength,time] = computeBug1(start,goal,stepSize,ObstaclesList)
currentPosition = start;
sequence = currentPosition;
pathLength = 0;
 T = 1;
% compute distance between each point
distance = @(p1,p2) norm(p1 - p2);
k=1;
int = 0;
Q = 0;
distanceArray = [];
timeArray = [];
tStart= tic;
stepSize3 = stepSize;
while distance(currentPosition, goal) > stepSize

    Q = Q + 1;
    minGoal(Q) = distance(currentPosition, goal);
    minGoalIndex = find(minGoal == min(minGoal));
    % Move directly towards the goal
    [0 L] = size(ObstaclesList);
    for i = 1:L
    q = currentPosition;
    Poly = cell2mat(ObstaclesList{i}');
    [~,Minseg,MinVertexDistance] = computeTangentVectorToPolygon(Poly,currentPosition);

    closestSegment(i) = Minseg;
    MinVertex(i) = MinVertexDistance;
    end
     closestVertexP = min(MinVertex);
     closestSegmentP = min(closestSegment);
     closeIndexV = find(MinVertex == closestVertexP);
```

```
     closeIndexS = find(closestSegment == closestSegmentP);
     ClosestOBC = cell2mat(ObstaclesList{closeIndexS}');
     plot(start(1,1),start(1,2),'Or');
    hold on
    plot(goal(1,1),goal(1,2),'Or');
    hold on
     P = ClosestOBC;
    q = currentPosition;
[U,~,~] = computeTangentVectorToPolygon(P,q);



   if closestSegmentP <= stepSize3 || closestVertexP <= stepSize


            nextPos = currentPosition + stepSize3 * U;
            PrevU(k,:) = round(U,2);
            sequence = [sequence; nextPos];
            currentPosition = nextPos;

            PositionAtOB(T,:) = currentPosition;
             pathLength = pathLength + stepSize;
            time = toc(tStart);
            timeArray = [timeArray, time];
            distanceArrayPlot = distance(currentPosition,goal);
            distanceArray = [distanceArray, distanceArrayPlot];
            T = T +1;


                    % plot
          figure(1)
         hold on
        plot(sequence(k,1),sequence(k,2),'Og')
```

```
              k = k +1;
                if closeIndexS == L && time > 4
                stepSize3 = 0.045;
               end
               if closestVertexP <= stepSize


               nextPos = currentPosition + stepSize3 *  PrevU(k-1,:) ;
               sequence = [sequence; nextPos];


               currentPosition = nextPos;

               PositionAtOB(T,:) = currentPosition;
               T = T +1;
                 Contact = k - T;
               int = 6;
                pathLength = pathLength + stepSize;
                      time = toc(tStart);
               timeArray = [timeArray, time];
                distanceArrayPlot = distance(currentPosition,goal);
               distanceArray = [distanceArray, distanceArrayPlot];

               % plot
             figure(1)
            hold on
           plot(sequence(k,1),sequence(k,2),'Og')

              k = k +1;

               end
               while int > 5
               if  distance(PositionAtOB(1,:), currentPosition) <= stepSize
                      ShortestPointToGoal = sequence(minGoalIndex,:);
                      for R = 1: minGoalIndex - Contact
```

```
                      sequence(k+R,:) = sequence(Contact+R,:);
                      currentPosition = sequence(k+R,:);

                pathLength = pathLength + stepSize;
                      time = toc(tStart);
               timeArray = [timeArray, time];
                distanceArrayPlot = distance(currentPosition,goal);
               distanceArray = [distanceArray, distanceArrayPlot];

               % plot
             figure(1)
            hold on
           plot(sequence(k+R,1),sequence(k+R,2),'O')
              k = k +1;

                      end
                       k = k +R;
                      while distance(ShortestPointToGoal,currentPosition) < 0.4

                       nextPos = currentPosition + stepSize * (goal - currentPosition) / distance(currentPosition, goal);
                     sequence = [sequence; nextPos];

                      pathLength = pathLength + stepSize;
                     currentPosition = nextPos;

                              time = toc(tStart);
               timeArray = [timeArray, time];
                distanceArrayPlot = distance(currentPosition,goal);
               distanceArray = [distanceArray, distanceArrayPlot];

               % plot
             figure(1)
            hold on
           plot(sequence(k,1),sequence(k,2),'Og')
```

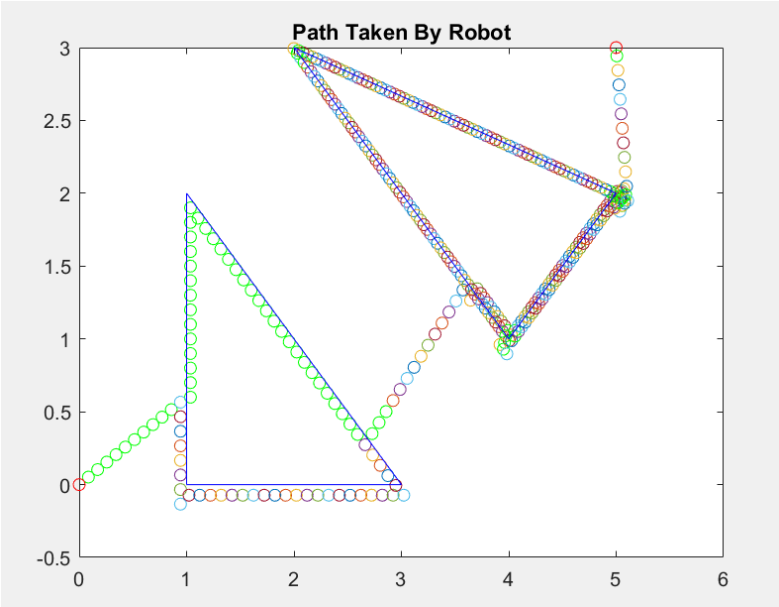
```
                k = k +1;
                        end

                        break
                 else
                    break
                 end

                 end

        else
            nextPos = currentPosition + stepSize * (goal - currentPosition) / distance(currentPosition, goal);
            sequence = [sequence; nextPos];

             pathLength = pathLength + stepSize;
            currentPosition = nextPos;
             time = toc(tStart);
             timeArray = [timeArray, time];
              distanceArrayPlot = distance(currentPosition,goal);
                  distanceArray = [distanceArray, distanceArrayPlot];
             % plot
                figure(1)
               hold on
              plot(sequence(k,1),sequence(k,2),'Og')

                k = k +1;

        end

    end
    hold off;
    figure(2)
    scatter(timeArray,distanceArray);
             end
```

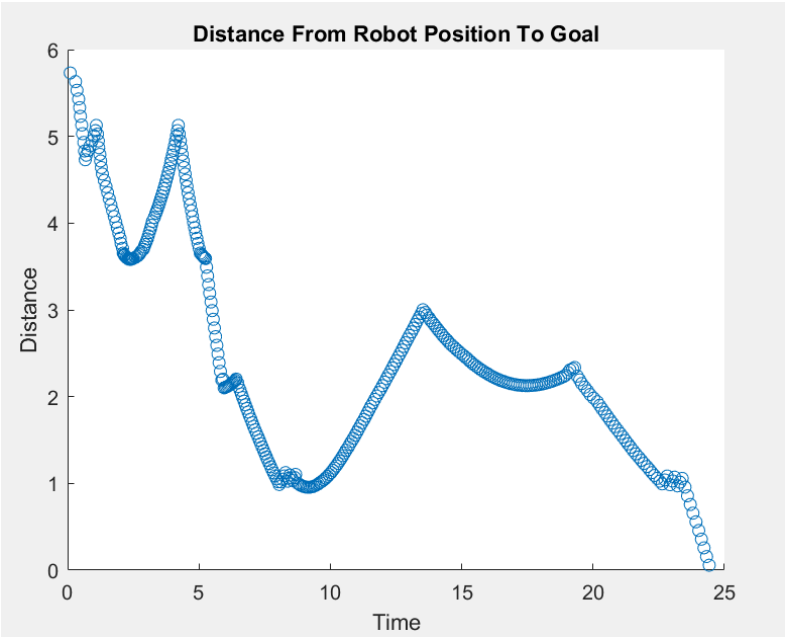
The code contains a while loop that checks if the distance from the current position to the goal is less than the step size to determine when the point has reached the goal. Inside the while loop, there are a couple of if statements that determine if the point has hit an obstacle or if the point has a clear path to the goal. To determine if the point has hit an obstacle the computeTangentVectorToPolygon function is used to determine the distance from the current position to the nearest segment and the nearest vertex. If either of these distances are less than the step size, then the point knows it's at an obstacle and will start traveling around the obstacle using the U vector output from the function. The sequence of this path taken by the point is recorded after every step and the minimum distance from the point to the goal is also recorded after every step to help determine the closest point of exit from the obstacle to the goal. The shortest point to the goal will be captured and the sequence of the point traveled will be repeated back to the shortest point to the goal. A while loop is then used to allow the point to follow the path towards the goal. The code also contains for-loops that help create necessary arrays. The step size is altered once the point reaches the second obstacle in the given example as this helped the point circumnavigate the obstacle. I was not able to have the point leave from the shortest distance to the goal off of the second obstacle.

iv) Given Example

Input: [sequence,pathLength,time] = computeBug1([0 0],[5 3],[0.1], {{[1, 2], [1, 0], [3, 0]}, {[2, 3], [4, 1],[5, 2]}})

Output:





Name	Value
pathLength	37.2000
sequence	*406x2 double*
time	24.4205