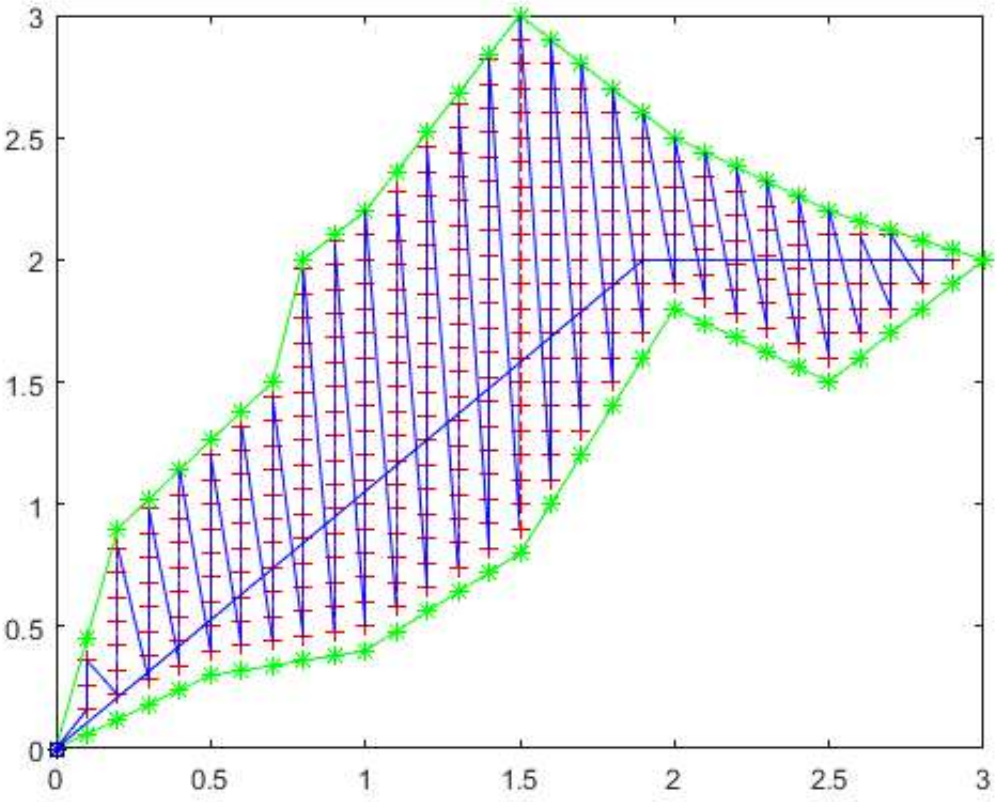
```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%              Mähstrategie (Matlab Simulation)                        %
%                                                                       %
% Autoren: Tom Niehaus, Florian Müller                                  %
%                                                                       %
% Änderungsdatum: 08.12.2016                                            %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [  ] = Maehstrategie( )
%Die Mähstrategie gibt vor wie das Rasenstück abgefahren wird.
%   Es wird davon ausgegangen, dass der Startpunkt an der Ladestation liegt

    clear all
    clc
    close all

    %Default Wert da die Funkten getAktuellePosition die vorherige
    %AktuellePosition übergeben bekommt
    aktuelle_Position = [0,0,1];

    %Ausrichtung nach Norden der internen Karte
    Ausrichtungsvektor = richteNachObenAus();


    %Die Karte wird eingelesen und in Karten_Punkte gespeichert
    [Karten_Punkte] = karte_aufzeichnen();



    hold on


    %Solange das letzte Element nicht gemäht wurde wird fortgesetzt
     while Karten_Punkte(3,size(Karten_Punkte,2)) == 0
        aktuelle_Position = getAktuelle_Position(Karten_Punkte, aktuelle_Position);
        naechste_Position = getNaechste_Position(Karten_Punkte, aktuelle_Position);
        [aktuelle_Position, Karten_Punkte] = fahreNaechstenPunktAn(aktuelle_Position, naec
hste_Position, Ausrichtungsvektor, Karten_Punkte);
     end

     fahrezurLadestation(aktuelle_Position, Karten_Punkte, Ausrichtungsvektor)


end


function [ aktuelle_Position, Karten_Punkte ] = fahreNaechstenPunktAn(aktuelle_Position, n
aechste_Position, Ausrichtungsvektor, Karten_Punkte)
%Der nächste Punkt wird angefahren
%   Der Roboter richtet sich aus und fährt solange die aktuelle_Position
%   nicht der naechsten Position entspricht


% Der Bewegungsvektor wird aus der nächsten sowie der aktuellen Position
% bestimmt sowie der Betrag
    Bewegungsvektor(1) = naechste_Position(1) - aktuelle_Position(1);
    Bewegungsvektor(2) = naechste_Position(2) - aktuelle_Position(2);
```

```
    Betrag_Ausrichtungsvektor = sqrt(Ausrichtungsvektor(1)^2 + Ausrichtungsvektor(2)^2);
    Betrag_Bewegungsvektor = sqrt(Bewegungsvektor(1)^2 + Bewegungsvektor(2)^2);


    Winkel_zur_NaechstenPosition_rad = acos((dot(Ausrichtungsvektor, Bewegungsvektor))/(Be
trag_Ausrichtungsvektor * Betrag_Bewegungsvektor));

    Winkel_zur_NaechstenPosition_deg = rad2deg(Winkel_zur_NaechstenPosition_rad);


    dreheUmWinkel(Winkel_zur_NaechstenPosition_deg);


     while aktuelle_Position(1) ~= naechste_Position(1) || aktuelle_Position(2) ~= naechst
e_Position(2)
         [aktuelle_Position, Karten_Punkte] = fahre(Betrag_Bewegungsvektor, Bewegungsvekto
r, aktuelle_Position,naechste_Position, Karten_Punkte);
     end
end

function [ aktuelle_Position, Karten_Punkte ] = fahre( Entfernung, Bewegungsvektor, aktuel
le_Position, naechste_Position, Karten_Punkte )
%Die nächste Übergebene Position wird angefahren

% Das Anfahren wird in der Simulation als Plot der Strecke in der Karte
% dargestellt
    plot([naechste_Position(1),aktuelle_Position(1)],[naechste_Position(2),aktuelle_Positi
on(2)] , 'b');

% Die vorherige Stelle wird als gemäht markiert und die nächste Position
% als Aktueller Standort
    Karten_Punkte(3, aktuelle_Position(3)) = 1;
    Karten_Punkte(3, naechste_Position(3)) = 3;
    aktuelle_Position = naechste_Position;

end

function [ Ausrichtungsvektor ] = richteNachObenAus()
%Zunächst richtet sich der Roboter nach oben aus um im Koordinatensystem in
%der internen Karte nach Norden zu schauen
%   Der Vektor entspricht der y-Achse

    Ausrichtungsvektor = [0, 1];

end

function [ aktuellePosition ] = getAktuelle_Position( Karten_Vektor, aktuellePosition )
%Die Aktuelle Position wird herausgefunden, wenn diese nicht bekannt ist,
%d.h nur bei Start des Mähens
%   Wenn die aktuelle Position nicht bekannt ist, wird die Karte
%   durchlaufen und in der dritten Zeile nach einer 3 gesucht (die 3 steht für die aktuell
e Position)
%   Die aktuelle Position wird dann mit den Positionen aus der Karte
%   beschrieben

    if Karten_Vektor(3, aktuellePosition(3)) ~= 3
        for i = aktuellePosition(3):1:length(Karten_Vektor)
            if Karten_Vektor(3,i) == 3
                aktuellePosition(1) = Karten_Vektor(1,i);
                aktuellePosition(2) = Karten_Vektor(2,i);
```

```
                aktuellePosition(3) = i;
            end
        end
    end

end


function [ naechste_Position ] = getNaechste_Position( Karten_Vektor, aktuelle_Position )
%Die nächste Position wird erkannt und ausgelesen
%   Die Karte wird durchlaufen und nach dem nächsten Element gesucht,
%   welches nicht gemäht ist (3. Zeile der Matrix = 0)


    for i = aktuelle_Position(3):1:length(Karten_Vektor)
      if Karten_Vektor(3,i) == 0
        naechste_Position(1) = Karten_Vektor(1,i);
        naechste_Position(2) = Karten_Vektor(2,i);
        naechste_Position(3) = i;
        break;
      end
    end


end


function [  ] = dreheUmWinkel( Winkel )
%dreht den Roboter um einen bestimmten Winkel
%


end

function [  ] = fahrezurLadestation( aktuelle_Position, Karten_Vektor, Ausrichtungsvektor
)
%Roboter fährt zur Ladestation
%   Roboter fährt zunächst in x-Richtung zur Ladestation und dann im
%   direkten Weg auf die Ladestation zu


% Alle Punkte der Karte auf gleicher Höhe werden gesucht und der
% Mittelpunkt dieser wird angefahren
    Hilfsvariable = 1;
    for i = 1:1:length(Karten_Vektor)
        if Karten_Vektor(2,i) == aktuelle_Position(2)
            Hilfsvariable_Mitte_anfahren(Hilfsvariable,1) = Karten_Vektor(1,i);
            Hilfsvariable_Mitte_anfahren(Hilfsvariable,2) = Karten_Vektor(2,i);
            k(Hilfsvariable) = i;
            Hilfsvariable = Hilfsvariable + 1;
        end
    end


    j = floor(Hilfsvariable/2);


% Die nächste Position in der Mitte der Karte wird in naechste_Position
% geschrieben
    naechste_Position(1) = Karten_Vektor(1, k(j));
    naechste_Position(2) = Karten_Vektor(2, k(j));
```

```
    naechste_Position(3) = k(j);


% Der neue Bewegungsvektor wird bestimmt sowie der Betrag des Vektors
    Bewegungsvektor(1) = naechste_Position(1) - aktuelle_Position(1);
    Bewegungsvektor(2) = naechste_Position(2) - aktuelle_Position(2);

    Betrag_Bewegungsvektor = sqrt(Bewegungsvektor(1)^2 + Bewegungsvektor(2)^2);


    Betrag_Ausrichtungsvektor = sqrt(Ausrichtungsvektor(1)^2 + Ausrichtungsvektor(2)^2);
    Betrag_Bewegungsvektor = sqrt(Bewegungsvektor(1)^2 + Bewegungsvektor(2)^2);

%Winkel wird berechnet und um den gewünschten Winkel gedreht
    Winkel_zur_NaechstenPosition_rad = acos((dot(Ausrichtungsvektor, Bewegungsvektor))/(Be
trag_Ausrichtungsvektor * Betrag_Bewegungsvektor));

    Winkel_zur_NaechstenPosition_deg = rad2deg(Winkel_zur_NaechstenPosition_rad);

    dreheUmWinkel(Winkel_zur_NaechstenPosition_deg);

% Die Position wird angefahren
    [aktuelle_Position, Karten_Vektor] = fahre(Betrag_Bewegungsvektor, Bewegungsvektor, ak
tuelle_Position,naechste_Position, Karten_Vektor);

% Der Roboter muss sich nun in Richtung der Ladestation bewegen daher entspricht die nächs
te Position [0, 0, 1]
    naechste_Position(1) = 0;
    naechste_Position(2) = 0;
    naechste_Position(3) = 1;

% Die Position wird angefahren
    [aktuelle_Position, Karten_Vektor] = fahre(Betrag_Bewegungsvektor, Bewegungsvektor, ak
tuelle_Position,naechste_Position, Karten_Vektor);


end
```



Published with MATLAB® R2016b