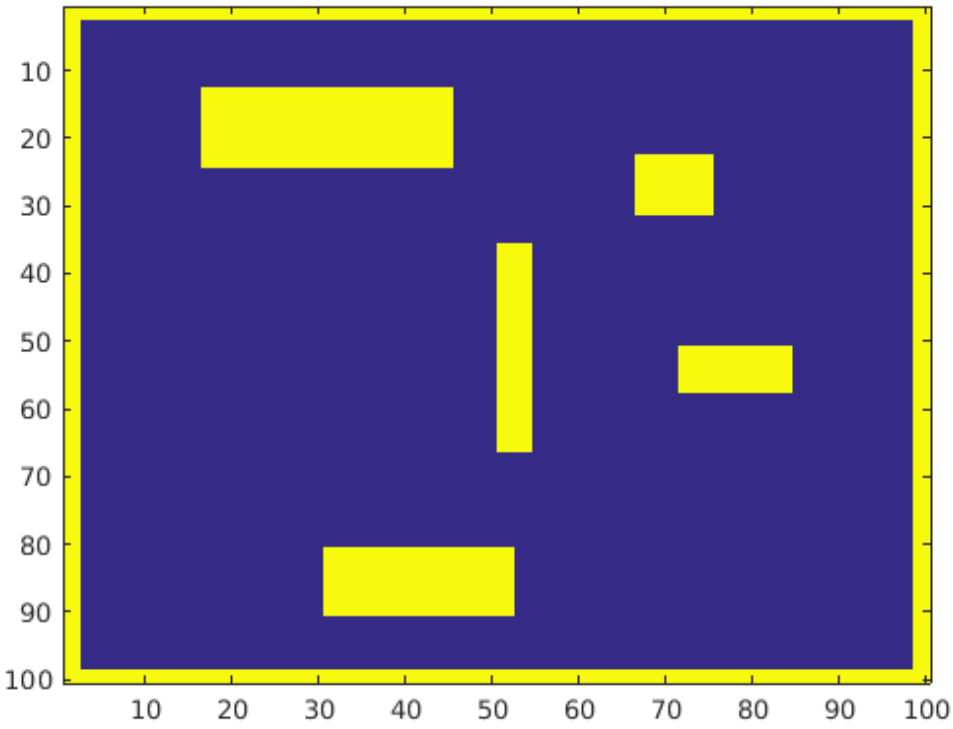
Table of Contents

SIMULATION PLATINE

P(i).s : P(i).s(t) est égal à la longueur du chemin parcouru par l'objet i au bout d'une durée t (t est entier positif) P(i).pp : P(i).pp(t) est égal à l'index permettant d'accéder à la position du centre de l'objet i à l'instant t : (x,y) = (xy(1,round(P(i).pp(t)),i),xy(2,round(P(i).pp(t)),i))

```
pasTemps=0.4;
%
xy = zeros(3,subd,nombre_objets); % xy(1:2,:,i) : echantillons du
 chemin i
xy2 = zeros(2,subd);
s1 = zeros(1,nombre_objets); % s1(i) : longueur du chemin i
%
%
hdepl = figure;
hdepl.Name = 'Simulation PLATINE';
hdepl.Units = 'centimeters';
imagesc(im);
hold on
%
```

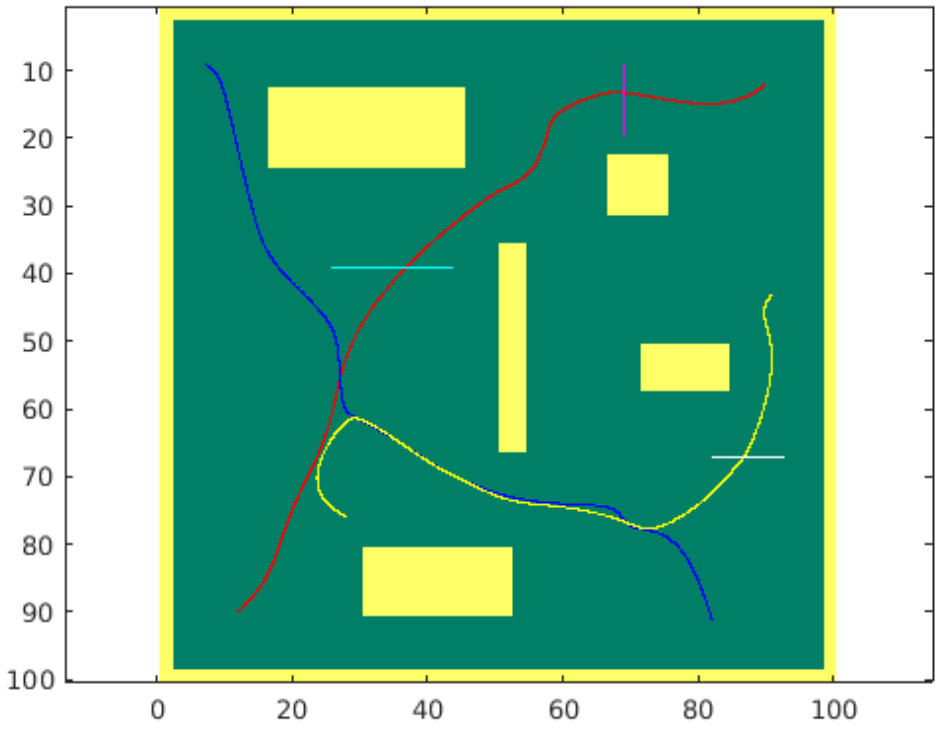


Echantillonnage des chemins, calcul de leurs longueurs et tracés de ces chemins

attention: on peut parcourir le même chemin plusieurs fois

```
couleur = char('Red' , 'Blue','Cyan','Magenta', 'Yellow', 'White');
length_couleur = [3 4 4 7 6 5];
%
for i = 1:nombre_objets
    if strcmp(T(i).chemin,'DROITE')
        % Le chemin est une droite
        idep = T(i).depart(1);
        jdep = T(i).depart(2);
        iarr = T(i).arrivee(1); % T(i).arrivee(i) n'est pas une
 coordonnee du pt d'arrivees si nbrerepetition est pair !!
        jarr = T(i).arrivee(2);
        %
        for p=1:subd
            mu = (p-1)/subd;
            xy2(2,p) = idep + mu*(iarr-idep);
            xy2(1,p) = jdep + mu*(jarr-jdep);
        end
        xy(1:2,:,i) = xy2;
        plot(xy2(1,:),xy2(2,:),couleur(i,:));
    else
```

```
        if strcmp(T(i).chemin,'NURBS')
            % Le chemin est une NURBS
            nurbsf = T(i).nurbs;
            xy(:,:,i) = nrbeval(nurbsf,linspace(0.0,1.0,subd));
            xy2 = xy(1:2,:,i); %inutile
            nrbplot(nurbsf,subd,couleur(i,:));
        else
            disp('ERREUR');
        end
    end
    % arclength: compute arc length of any curve represented as a
 sequence of points
    s1(i) =arclength(xy(1,:,i),xy(2,:,i)); % calcul longueur chemin i
end
```

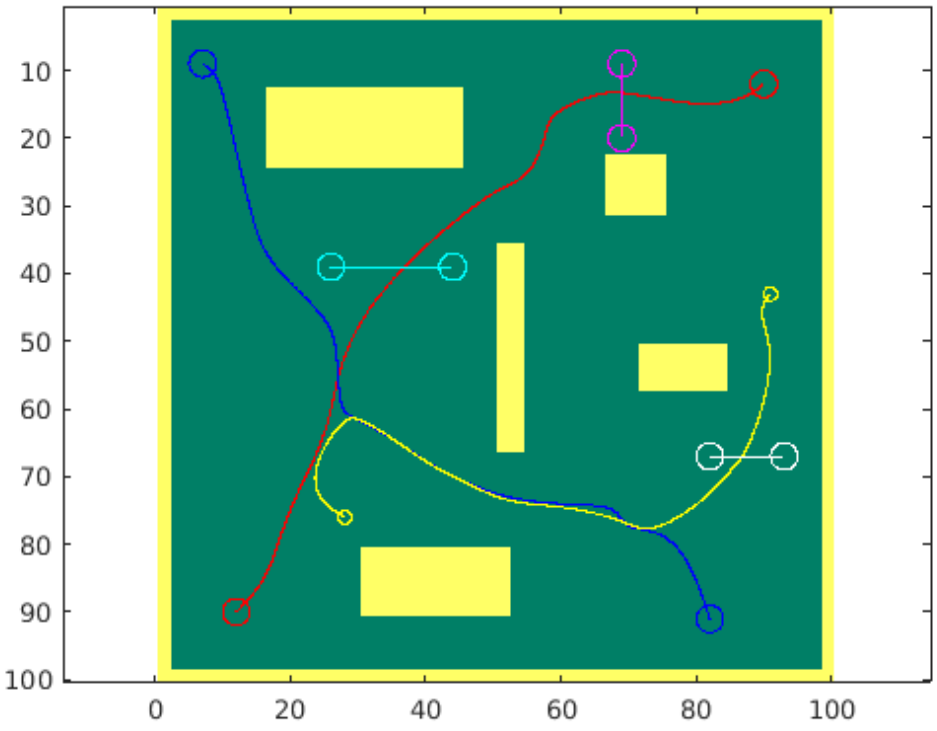


Tracés des positions de départ et d'arrivée des objets

Visualiser le robot a son point de depart

```
th = 0:3.14/50:2*3.14; % angles pour tracer un cercle
lth = length(th);
figure(hdepl);
%
for i = 1:nombre_objets
```

```
    % Visualisation position de depart de l'objet i
    idep = T(i).depart(1);
    jdep = T(i).depart(2);
    diametre_robot = T(i).diametre_robot;
    %
    xunit = double(idep) + diametre_robot * cos(th);
    yunit = double(jdep) + diametre_robot * sin(th);
    hold on;
    depart=plot(yunit, xunit,couleur(i,:)); % image -> figure  (x,y) -
> (y,x)
    %
    %  Visualisation position d'arrivee de l'objet i
    iarr = T(i).arrivee(1);
    jarr = T(i).arrivee(2);
    %
    xunit = double(iarr) + diametre_robot * cos(th);
    yunit = double(jarr) + diametre_robot * sin(th);
    hold on;
    arrivee=plot(yunit, xunit, couleur(i,:)); % image -> figure  (x,y)
 -> (y,x)
end
```



Calcul des trajectoires (chemin + cinématique)

```
rrob = zeros(1,nombre_objets);
for k = 1:nombre_objets
```

```
    rrob(k) = T(k).diametre_robot/2;
end
%
% Reindexation des chemins selon les abscisses curvilignes s(t)
p=1:subd;
%
tempsparcours = zeros(1,nombre_objets);
ii = zeros(1,nombre_objets);
%
for i=1:nombre_objets
    v = T(i).vitesse; % vitesse de l'objet i égal à v en unités pixels
    % La cinématique est donnée par P(i).s et par P(i).pp
    % 1. Calcul de P(i).s
    s=0:v:s1(i); % s = 0 v 2v 3v ... s1(i) donc delta(s) = v donc
 vitesse constante v
    xy2 = xy(1:2,:,i);
    pp0 = pdearcl(p,xy2,s,0,s1(i)); % reindexation du chemin i selon
 le vecteur d'abscisse curviligne s
    %
    nbrerepetition = T(i).nbre_repetition;
    nbrefois = nbrerepetition - 1;
    %
    s = 0:v:nbrerepetition*s1(i);
    P(i).s = s;
    tempsparcours(i) = length(s); % temps du parcours de l'objet i (si
 pas de collision)
    T(i).tempsparcours = tempsparcours(i);
    %
    % 2. Calcul de P(i).pp
    pp1 = pp0;
    for k=1:nbrefois
        pp00 = pp1(end-1:-1:1); % correction end devient end-1
 29/11/2016
        pp0 = [pp0 pp00];
        pp1 = pp00;
    end
    P(i).pp = pp0; % sauvegarde de pp0 dans structure P
end
```

SIMULATION des deplacements des objets

```
Temps_reparation1 = T(1).temps_repar; % Temps de reparation de l'objet
 1
portee = T(1).portee; % horizon du robot 1
Nombre_reparations = 0;
detection = false(1,nombre_objets);
% detection(i) == 1 si l'objet 1 a detecte l'objet i
% i.e. l'objet i s'est approche de l'objet 1 d'une distance inférieure
 à
% portee
maxtempsparcours = max(tempsparcours);
%
t = 1;
```

```
objets_heurtes = [];
while t <= maxtempsparcours % Boucle temporelle avec maxtempsparcours
 variable à cause des collisions éventuelles
    %
    xunit = [];
    yunit = [];
    if exist('robot','var') == 1
        delete(robot);
    end
    for k = 1:nombre_objets
        if t < tempsparcours(k)
            ii(k) = round(P(k).pp(t)); %% ii(k) est l'index donnant
 accès à la position du centre de l'objet k à l'instant t
            xcentre = xy(1,ii(k),k);
            ycentre = xy(2,ii(k),k);
        else % t >= tempsparcours(k)
            if t == tempsparcours(k)
                ii(k) = round(P(k).pp(t)); %% ii(k) est l'index
 donnant accès à la position du centre de l'objet k à l'instant
 tempsparcours(k)
                % L'objet k est arrive a destination !
                mem = length_couleur(k);
                cprintf(couleur(k,1:mem),['Arrivee de l''objet
 ',num2str(k), ' au bout de ',num2str(t),' secondes']);
                disp(' ');
            end
            if strcmp(T(k).chemin,'DROITE') &&
 (mod(T(k).nbre_repetition,2)==1)
                xcentre = T(k).depart(2);
                ycentre = T(k).depart(1);
            else
                xcentre = T(k).arrivee(2);
                ycentre = T(k).arrivee(1);
            end
        end
        xunit0 = xcentre + rrob(k) * cos(th);
        yunit0 = ycentre + rrob(k) * sin(th);
        xunit = [xunit xunit0];
        yunit = [yunit yunit0];
    end
    affich_robots; %affichage des objets
    drawnow;
    pause(pasTemps);
    %
    % Test de détection d'obstacles de l'objet 1 avec les autres
 objets
    % kobj
    % On ne considère ici que les collisions de l'objet 1 avec les
 autres
    % objets
    %
    for kobj = 2:nombre_objets
        % kobj est le numero de l'objet susceptible de rentrer en
 collision avec l'objet 1
```

```
        distance = norm(xy(1:2,ii(kobj),kobj)-xy(1:2,ii(1),1)); %
 distance du robot (objet 1) à l'objet kobj
        sumray = rrob(1)+rrob(kobj);
        if distance < sumray && ~any(ismember(objets_heurtes,kobj))
            % collision de l'objet 1 avec l'objet kobj à l'instant t!!
            % car la distance les séparant est trop petite et ils ne
 sont
            % jamais entres en collision precedemment
            % l'objet 1 ne pourra plus rentrer desormais en collision
 avec
            % kobj
            objets_heurtes = [objets_heurtes kobj];
            kcol = kobj; % on sauvegarde le numero de l'objet kobj
 rentré en collision avec l'objet 1
            load gong.mat
            sound(y)
            cprintf([1,0,0],['COLLISION AVEC OBJET
 ',num2str(kobj), ' !! au bout de ',num2str(t),' secondes']);
            disp(' ');
            % Calcul des longueurs parcourues par chaque objet i au
 moment
            % de la collision
%             for i = 1:nombre_objets
%                 longueur = P(i).s(t);
%                 cprintf([0,0,1],['Longueur parcourue par objet
 ',num2str(i),' : ', num2str(longueur),' pixels']);
%                 disp(' ');
%             end
            pause(1);
            % Il y a eu collision de l'objet 1 avec l'objet kcol,
            % on répare l'objet 1
            Nombre_reparations = Nombre_reparations + 1;
            cprintf([0,0,0],['Temps de reparation:
 ',num2str(Temps_reparation1),' ']);
            disp(' ');
            %
            % On recalcule les trajectoires des objets 1 et kcol
 entrés en collision,
            % les chemins restant les mêmes; les autres trajectoires
            % restent inchangées
            %
            num = [1 kcol];
            for i = num(1):num(end)
                % 1. MAJ de P(i).s
                savantcollision = P(i).s(1:t);
                Temps_reparation = T(i).temps_repar;
                ss = P(i).s(t) * ones(1,Temps_reparation);
                saprescollision = P(i).s(t+1:end);
                s = [savantcollision , ss , saprescollision];
                P(i).s = s;
                % 2. MAJ de P(i).pp
                pavantcollision = P(i).pp(1:t);
                ppp = P(i).pp(t) * ones(1,Temps_reparation);
                paprescollision = P(i).pp(t+1:end);
```

```
                P(i).pp= [pavantcollision , ppp , paprescollision];
                % 3. MAJ de tempsparcours(i)
                tempsparcours(i) = tempsparcours(i) +
 Temps_reparation;
                T(i).tempsparcours = tempsparcours(i);
            end
            maxtempsparcours = max(tempsparcours); % reactualisation
 du temps maximal de parcours
        else %pas de collision : distance >= sumray
            if distance <= portee % sumray <= distance <= portee
                % l'obstacle kobj est detecte
                % Etait-il deja detecte a l'instant precedent ?
                % si non on ecrit le message objet detecte
                if ~ detection(kobj)
                    cprintf([1,0.5,0],['OBJET ',num2str(kobj),'
 DETECTE']);
                    disp(' ');
                    detection(kobj) = true;
                end
            else % distance > portee
                % Si l'objet etait precedemment detecte on ecrit le
 message
                % fin de detection
                if detection(kobj)
                    cprintf([0,0,0],['fin detection objet
 ',num2str(kobj)]);
                    disp(' ');
                    detection(kobj) = false;
                end
            end
        end
    end
    %
    t = t + 1;
end
```

```
Arrivee de l'objet 6 au bout de 11 secondes
OBJET 5 DETECTE

Index exceeds matrix dimensions.

Error in Simulation3 (line 143)
            ii(k) = round(P(k).pp(t)); %% ii(k) est l'index donnant
 accès à la position du centre de l'objet k à l'instant t
```

Durée totale écoulée

```
tf = t-1;
cprintf('Blue',['Durée de la mission: ',num2str(tf),' secondes']);
disp(' ');
```

Published with MATLAB® R2015a