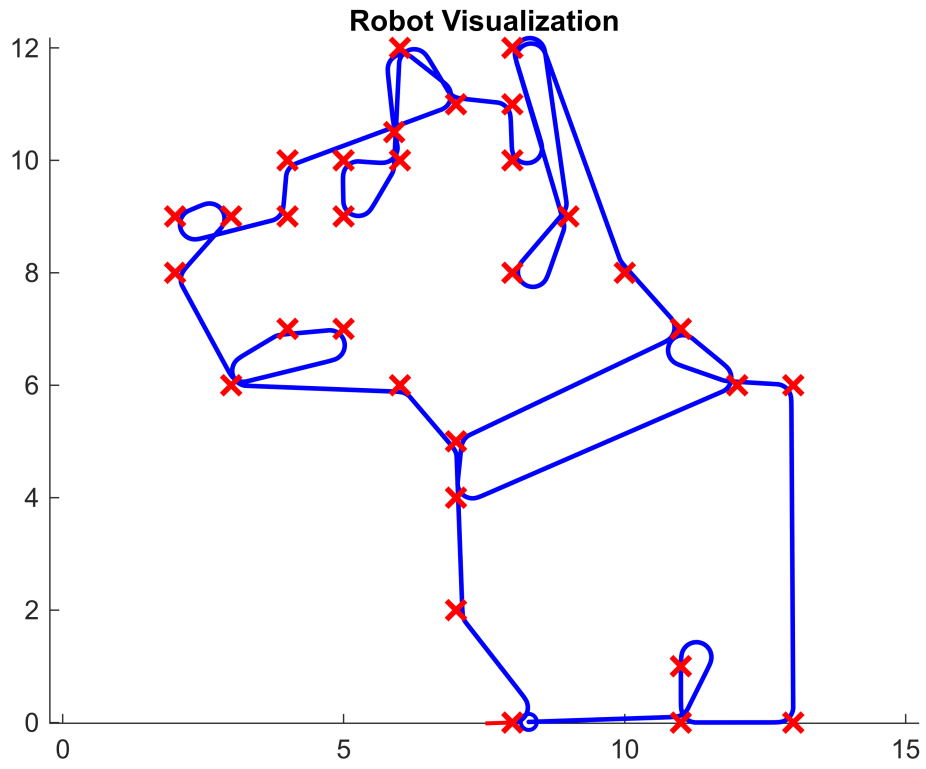
```
%% EXAMPLE: Differential drive vehicle following waypoints using the
% Pure Pursuit algorithm
%
% Copyright 2018-2019 The MathWorks, Inc.

%% Define Vehicle
R = 0.05;
L = 0.18;
dd = DifferentialDrive(R,L);

%% ======== SIMULACIÓN 1: SILUETA DE PERRO ========
% sampleTime = 0.05: Se utiliza un tiempo de muestreo pequeño para tener mayor
resolución y precisión
% en las curvas del contorno del perro, ya que hay muchos puntos y detalles
angulares que requieren
% seguimiento cercano.
%
% tVec = 0:sampleTime:220: Se extiende el tiempo total de la simulación para que el
robot tenga
% suficiente tiempo de recorrer toda la figura con velocidad baja sin quedar
incompleta.
%
% initPose = [8; 0; 0]: El robot comienza desde la base del cuello, justo en la
parte inferior de la
% figura del perro, mirando hacia arriba (ángulo 0 radianes).
%
% waypoints: Se definen a mano todos los puntos clave del contorno del perro,
incluyendo los detalles
% del hocico, orejas, cuello y cuerpo. Algunos puntos se repiten para permitir
trayectorias cerradas
% o dar forma a curvas más complejas..
sampleTime = 0.05;
tVec = 0:sampleTime:220;
initPose = [8; 0; 0];
pose = zeros(3,numel(tVec));
pose(:,1) = initPose;

waypoints = [
    7,2;
    7,5;
    6,6;
    3,6;
    4,7;
    5,7;
    3,6,
    2,8;
    3,9;
    2,9;
    4,9;
    4,10;
```

```
    7,11;
    6,12;
    5.9,10.5;
    6,10;
    5,10;
    5,9;
    6,10;
    6,12;
    7,11;
    8,11;
    8,10;
    8,12;
    9,9;
    8,8;
    9,9;
    8,12;
    10,8;
    11,7;
    7,5;
    7,4;
    12,6;
    11,7;
    12,6;
    13,6;
    13,0;
    11,0;
    11,1;
    11,0;
    8,0;
];

figure(1)
viz1 = Visualizer2D;
viz1.hasWaypoints = true;

controller = controllerPurePursuit;
controller.LookaheadDistance = 0.3;
controller.DesiredLinearVelocity = 0.4;
controller.MaxAngularVelocity = 1.5;

close(1)
r = rateControl(1/sampleTime);
currentIdx = 1;
goalRadius = 0.3;

for idx = 2:numel(tVec)
    if currentIdx > size(waypoints,1)
        break;
    end
    controller.Waypoints = [pose(1:2,idx-1)'; waypoints(currentIdx,:)];
```

```
    [vRef,wRef] = controller(pose(:,idx-1));
    [wL,wR] = inverseKinematics(dd,vRef,wRef);
    [v,w] = forwardKinematics(dd,wL,wR);
    velB = [v;0;w];
    vel = bodyToWorld(velB,pose(:,idx-1));
    pose(:,idx) = pose(:,idx-1) + vel*sampleTime;
    distance = norm(pose(1:2,idx) - waypoints(currentIdx,:)');
    if distance < goalRadius
        currentIdx = currentIdx + 1;
    end
    viz1(pose(:,idx),waypoints)
    waitfor(r);
end
```

```
Warning: System Object 'Visualizer2D' is inherited from mixin class 'matlab.system.mixin.CustomIcon' that
will no longer be supported. Remove 'matlab.system.mixin.CustomIcon' and define corresponding System object
methods instead.
```

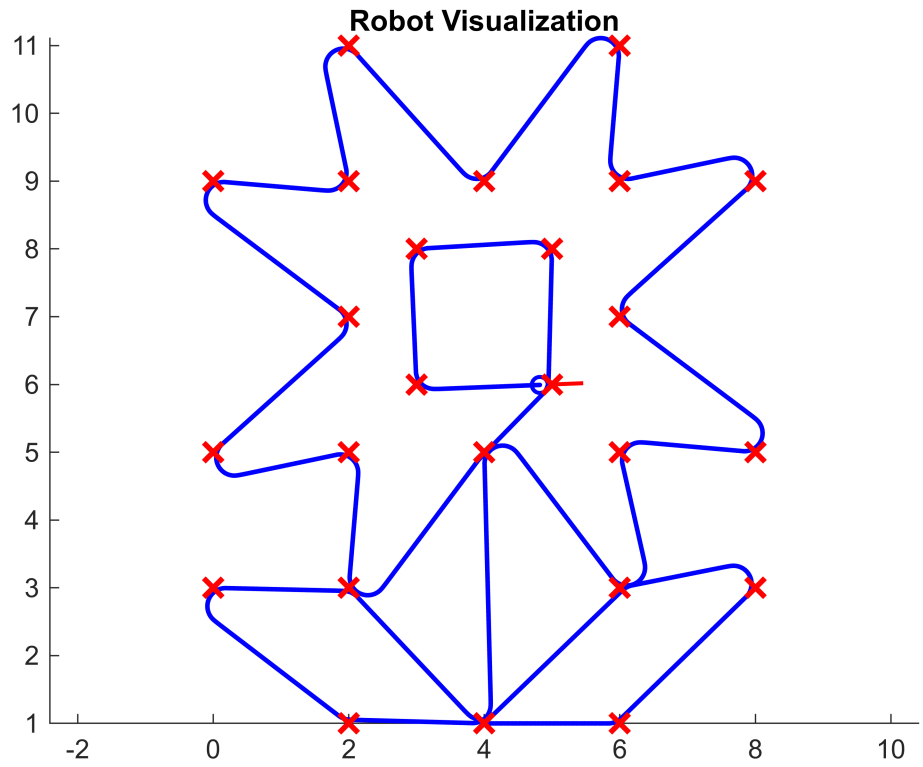



```
%% ======== SIMULACIÓN 2: FLOR ========
% sampleTime2 = 0.05: El tiempo de muestreo se mantiene pequeño para garantizar
precisión al seguir
% las curvas y esquinas de los pétalos de la flor. Este valor permite al robot
tener control refinado
% sobre los giros cerrados que se forman entre pétalos.
%
% tVec2 = 0:sampleTime2:250: Se da un tiempo amplio de simulación porque la flor
tiene muchas transiciones
```

```
% y trayectorias cerradas. Así se asegura que pueda completar la figura aunque
avance despacio.
%
% initPose2 = [4; 1; 0]: El robot empieza en la base del tallo (parte inferior
central), lo cual es
% conveniente porque permite avanzar hacia los pétalos siguiendo un recorrido
natural.
%
% waypoints2: Incluye los puntos del tallo, el centro de la flor y los pétalos. Se
repiten varios
% segmentos para asegurar que el robot recorra tanto las partes internas como
externas.
sampleTime2 = 0.05;
tVec2 = 0:sampleTime2:250;
initPose2 = [4; 1; 0];  % por ejemplo, esquina inferior izquierda
pose2 = zeros(3,numel(tVec2));
pose2(:,1) = initPose2;

% Puedes cambiar esto por la figura que desees
waypoints2 = [
   6,1;
   8,3;
   6,3;
   4,1;
   2,3;
   0,3;
   2,1;
   4,1;
   4,5;
   6,3;
   6,5;
   8,5;
   6,7;
   8,9;
   6,9;
   6,11;
   4,9;
   2,11;
   2,9;
   0,9;
   2,7;
   0,5;
   2,5;
   2,3;
   4,5;
   5,6;
   5,8;
   3,8;
   3,6;
   5,6;
```

```
];

figure(2)
viz2 = Visualizer2D;
viz2.hasWaypoints = true;

controller = controllerPurePursuit;
controller.LookaheadDistance = 0.25;
controller.DesiredLinearVelocity = 0.4;
controller.MaxAngularVelocity = 1.5;

close(2)
r2 = rateControl(1/sampleTime2);
currentIdx2 = 1;
goalRadius2 = 0.2;

for idx = 2:numel(tVec2)
    if currentIdx2 > size(waypoints2,1)
        break;
    end
    controller.Waypoints = [pose2(1:2,idx-1)'; waypoints2(currentIdx2,:)];
    [vRef,wRef] = controller(pose2(:,idx-1));
    [wL,wR] = inverseKinematics(dd,vRef,wRef);
    [v,w] = forwardKinematics(dd,wL,wR);
    velB = [v;0;w];
    vel = bodyToWorld(velB,pose2(:,idx-1));
    pose2(:,idx) = pose2(:,idx-1) + vel*sampleTime2;
    distance = norm(pose2(1:2,idx) - waypoints2(currentIdx2,:)');
    if distance < goalRadius2
        currentIdx2 = currentIdx2 + 1;
    end
    viz2(pose2(:,idx),waypoints2)
    waitfor(r2);
end
```

```
Warning: System Object 'Visualizer2D' is inherited from mixin class 'matlab.system.mixin.CustomIcon' that
will no longer be supported. Remove 'matlab.system.mixin.CustomIcon' and define corresponding System object
methods instead.
```




```
%% ======== SIMULACIÓN 3: CEREZA ========
% sampleTime3 = 0.04: Se usa un tiempo de muestreo aún menor porque la figura de la
cereza contiene
% trayectorias más curvas y con transiciones entre círculos (las frutas) y tallos
delgados. Esta
% resolución permite suavizar los giros.
%
% tVec3 = 0:sampleTime3:300: Se otorga un tiempo largo ya que la figura incluye dos
círculos, una rama
% y un tallo, con múltiples idas y vueltas. Para que el robot pueda recorrerlo
lentamente y con precisión,
% este tiempo es necesario.
%
% initPose3 = [7.8; 6.2; 0]: El robot inicia justo en el punto de unión entre la
rama y una de las cerezas,
% permitiendo avanzar con fluidez por el borde de las frutas y luego subir a la
rama.
%
% waypoints3: Esta secuencia representa dos círculos (las cerezas), su conexión por
medio del tallo y
% una hoja superior. Se incluyen repeticiones y transiciones suaves para que el
robot dibuje el contorno
% correctamente.
sampleTime3 = 0.04;
tVec3 = 0:sampleTime3:300;
```

```
initPose3 = [7.8; 6.2; 0];  % Posición inicial sugerida
pose3 = zeros(3,numel(tVec3));
pose3(:,1) = initPose3;

% Puedes modificar esta figura a tu gusto
waypoints3 = [
    9,5;
    9,3;
    7,1;
    5,1;
    3,3;
    3,5;
    5,7;
    7,7;
    7.8,6.2;
    10,9;
    8,11;
    6,11;
    4,9;
    10,9;
    7.8,6.2;
    7,5;
    6,6;
    7,5;
    8,5
];

figure(3)
viz3 = Visualizer2D;
viz3.hasWaypoints = true;

controller = controllerPurePursuit;
controller.LookaheadDistance = 0.2;
controller.DesiredLinearVelocity = 0.35;
controller.MaxAngularVelocity = 1.6;

close(3)
r3 = rateControl(1/sampleTime3);
currentIdx3 = 1;
goalRadius3 = 0.2;

for idx = 2:numel(tVec3)
    if currentIdx3 > size(waypoints3,1)
        break;
    end
    controller.Waypoints = [pose3(1:2,idx-1)'; waypoints3(currentIdx3,:)];
    [vRef,wRef] = controller(pose3(:,idx-1));
    [wL,wR] = inverseKinematics(dd,vRef,wRef);
    [v,w] = forwardKinematics(dd,wL,wR);
    velB = [v;0;w];
```

```
    vel = bodyToWorld(velB,pose3(:,idx-1));
    pose3(:,idx) = pose3(:,idx-1) + vel*sampleTime3;
    distance = norm(pose3(1:2,idx) - waypoints3(currentIdx3,:)');
    if distance < goalRadius3
        currentIdx3 = currentIdx3 + 1;
    end
    viz3(pose3(:,idx),waypoints3)
    waitfor(r3);
end
```

```
Warning: System Object 'Visualizer2D' is inherited from mixin class 'matlab.system.mixin.CustomIcon' that
will no longer be supported. Remove 'matlab.system.mixin.CustomIcon' and define corresponding System object
methods instead.
```