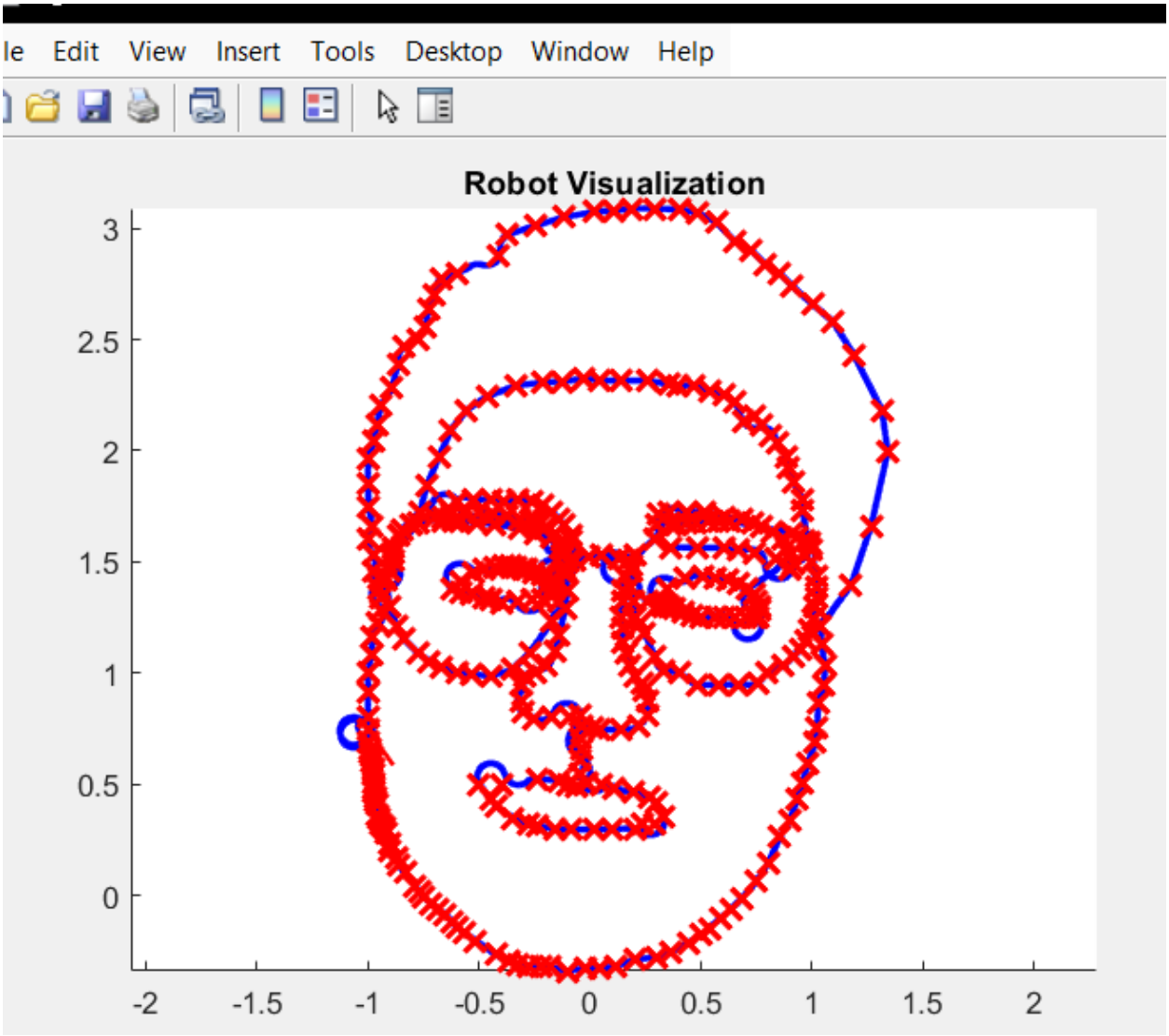
Evaluación 9 (Planeación de Trayectorias)

Para esta actividad utilicé la siguiente foto

Al final me decidí por un control de posición con lazo cerrado y con posiciones deseadas dadas ya que podía utilizar los waypoints de forma más sencilla y había menos parámetros que editar que con purse pursuit por ejemplo. Además de que necesitaba precisión milimétrica cosa que con purse pusuit no pude obtener fácilmente debido a que este solo calcula la curva hacia

Resultados



Si vemos la escala podemos darnos cuenta de que los errores al salirse de la trayectoria en curvas muy pronunciadas en realidad son mínimos y podemos darnos cuenta de que cumple con creces la expectativa de seguir la trayectoria del rostro sin errores grandes. Los

posibles aspectos para mejorar justamente es la capacidad del robot a dar giros muy pronunciados en poco tiempo.

```
clear
close all
clc
% Define Vehicle
R = 0.1; % Wheel radius [m]
L = 0.5; % Wheel base [m]
dd = DifferentialDrive(R, L);

% Simulation parameters
sampleTime = 0.00025;                    % Sample time [s]
tVec = 0:sampleTime:175.5;             % Time array

initPose = [-0.9960; 0.6910; deg2rad(280)];     % Initial pose (x, y, theta in
radians)
pose = zeros(3, numel(tVec));       % Pose matrix
pose(:,1) = initPose;

load waypoints_total.txt
waypoints = waypoints_total;

% Visualizer
viz = Visualizer2D;
viz.hasWaypoints = true;

% Pure Pursuit Controller
controller = controllerPurePursuit;
controller.Waypoints = waypoints;
controller.LookaheadDistance = 0.01;
controller.DesiredLinearVelocity = 0.185;
controller.MaxAngularVelocity = 2.97;

% Simulation loop
r = rateControl(1/sampleTime);
for idx = 2:numel(tVec)
    [vRef, wRef] = controller(pose(:,idx-1));
    [wL, wR] = inverseKinematics(dd, vRef, wRef);

    % Compute velocities
    [v, w] = forwardKinematics(dd, wL, wR);
    velB = [v; 0; w];                    % Velocities in body frame
    vel = bodyToWorld(velB, pose(:,idx-1)); % Transform to world frame

    % Discrete integration step
    pose(:,idx) = pose(:,idx-1) + vel * sampleTime;
end

% Visualización de trayectoria
viz = Visualizer2D;
viz.hasWaypoints = true;
```

```
for idx = 1:100:numel(tVec)  % Salta cada 10 muestras para visualización más fluida
    viz(pose(:,idx), waypoints);
    pause(0.00001);
end
```