```
function U = pointTrackingControl(X_toTrack, X_est, trackingThresholds)

    % Some useful variables
    K_rho = 1.0;
    K_alpha = 2.0;
    K_beta = -0.99;
    robotRadius = 0.2;
    robotWidth = robotRadius*2;
    wheelRadius = 0.1;

    % Some helpful calculations
    deltaX = X_toTrack(1) - X_est(1);
    deltaY = X_toTrack(2) - X_est(2);
    ang = atan2(deltaY,deltaX);


     % Calculate rho, the distance to goal
    rho = sqrt( deltaX^2 + deltaY^2 );

    % Check to see if simply rotation on the spot, if rho is small
    if rho < trackingThresholds(1)
        rho = 0;
        alpha = 0;
        beta = angleDiff(X_est(3) - X_toTrack(3));

    % if rho is still too big, invoke the point tracking theory
    else
        alpha = angleDiff(-X_est(3) + ang);
        beta = angleDiff(angleDiff(-X_est(3) - alpha) + X_toTrack(3));
    end

    % Calculate v, w
    v = K_rho * rho;
    w = K_alpha * alpha + K_beta * beta;

    % calculate in m/s, convert to rad/s
    desiredWheelSpeedR = (v + w*robotWidth)*(2*pi)/(2*pi*wheelRadius);
    desiredWheelSpeedL = (v -
w*robotWidth)*(2*pi)/(2*pi*wheelRadius);%was -ve

    % Check that max velocity in rad/s isn't reached
    currentMaxVel = max(abs(desiredWheelSpeedL),
abs(desiredWheelSpeedR));
    maxWheelSpeed = 2*pi;
    if  currentMaxVel > maxWheelSpeed
        desiredWheelSpeedR = desiredWheelSpeedR * maxWheelSpeed /
currentMaxVel;
        desiredWheelSpeedL = desiredWheelSpeedL * maxWheelSpeed /
currentMaxVel;
    end

    % Set the control vector U
    U = [desiredWheelSpeedR desiredWheelSpeedL];

end
```