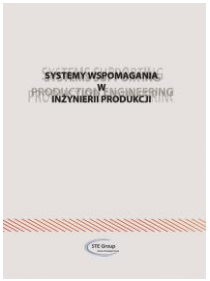
Development of a mobile robot control systemtitle of the paper

Data wpłynięcia do Redakcji: 06/2026
Data akceptacji przez Redakcję do publikacji: 07/2026

2026, volume 15, issue 1, pp. 78-91



Ivan Zajacko, Milan Saga
Zuzana Sagova, Pavol Bozek
University of Žilina**, Slovakia**

Abstract: The article discusses the issues of designing and developing a control system for a high-payload mobile robot (MR). Based on the technical assignment for the final qualifying work, requirements for the system were formulated: robot mass 100 kg, payload 100 kg, maximum speed 2 m/s. A control system architecture is proposed, including navigation, positioning, and obstacle detection subsystems. The trilateration method based on active ultrasonic beacons with digital temperature compensation of the speed of sound was used to determine coordinates, and a combination of incremental encoders and an electronic compass was used for local orientation. The motion control and obstacle avoidance algorithms are described. The results of modeling and polygon tests confirm the possibility of achieving positioning accuracy of ±10 cm and motion stability under full load.

Keywords: Mobile robot, Control system, Navigation, Trilateration, Ultrasonic sensors, Electronic compass, Obstacle avoidance Algorithm, Temperature compensation

INTRODUCTION

Autonomous mobile robots are widely used in logistics, industry, and the service sector. The key problem in creating such systems is ensuring reliable navigation and control in conditions of environmental uncertainty. The task of developing control systems for significant mass robots is especially relevant, where the inertial properties of the object impose strict requirements on the speed of control loops and positioning accuracy (Ivanov et al., 20220).

The goal of this work is to develop a mobile robot control system capable of autonomously moving to a specified goal, determining its coordinates, detecting and bypassing obstacles. The work is carried out within the framework of a technical assignment for a final qualifying work, where specific mass-dimensional and speed characteristics of the control object are specified.

ANALYSIS OF REQUIREMENTS AND EXISTING SOLUTIONS

According to the technical assignment, the control object is a mobile robot with a mass of 100 kg and a payload of 100 kg. The maximum movement speed is 2 m/s, maximum acceleration is 1 m/s^2. The engine type is electric. The system must provide the following functions.

Contact address: pavol.bozek@fstroj.uniza.sk

A. Movement of the MR to the specified goal;
B. Determination of the MR position coordinates;
C. Obstacle detection and bypass;
D. Submission of sound and light signals upon obstacle detection;
E. Formation of control signals for engines and reading of feedback signals.

Literature analysis shows that the following methods are most common for solving local orientation tasks:

A. Odometry. Use of incremental wheel rotation angle sensors (encoders). Allows tracking movement relative to the starting point, but is subject to error accumulation due to wheel slippage (Vlasov et al., 2017).
B. Beacon Systems. Use of stationary beacons (Ultrasonic, Infrared, Radio). The trilateration method allows determining absolute coordinates in local space. Ultrasonic systems are safe and inexpensive but require accounting for the environment temperature to correct the speed of sound (Galieva et al., 2024).
C. Computer Vision Systems and Laser Scanners (SLAM). Provide high accuracy in map construction, but require significant computational resources and are sensitive to lighting and dust conditions (Safin et al., 2023).

For this project, a combined approach was chosen: odometry for short-term speed tracking and an ultrasonic positioning system for correcting global coordinates. The choice is justified by safety requirements (absence of laser radiation), cost, and stability to industrial workshop conditions (Fukumura et al., 2024).

MATERIALS AND METHODS

The control system is built on a hierarchical principle, including upper (strategic) and lower (executive) levels.

Lower Control Level

The lower level implements control loops for electric motor rotation speed. Incremental encoders installed on the motor shafts are used to measure speed. The transfer function of the engine, taking into account the reducer and load, is described in state space. To ensure motion stability with a mass of 200 kg (own weight + load), PID regulators are used, whose coefficients are tuned taking into account speed requirements (acceleration 1 m/s^2) (Vu et al., 2025). Feedback on motor current allows indirect assessment of the load and detection of mechanical obstacles at an early stage (Fedorov et al., 2021; Jhong et al., 2023; Meshcheryakova et al., 2024; Suarez-Rivera et al., 2021; Phuong et al., 2023; Raikwar et al., 2022; Lee et al., 2022; Shi et al., 2023; Tuleshov et al., 2025; Leiva et al., 2025; Bandyopadhyay et al., 2021; Nikitin et al., 2022; Božek et al., 2021).

Positioning Subsystem

A system of active ultrasonic beacons is used to determine absolute coordinates. The principle of operation is based on measuring the time of signal passage from the beacon to the robot. The distance is calculated by the formula:

$$D=c\cdot t/2 \quad (1)$$

where:

c is the speed of sound in the medium, m/s (at 20°C in air ≈ 343 m/s);

t is the signal flight time from the transmitter to the receiver,

s; d is the distance from the transmitter to the center of the beacon and from the receiver to the center of the MR (Galieva et al., 2024).

After measuring distances to three or more beacons with known coordinates (xi,yi,zi), the robot position (x,y,z) is calculated by the trilateration method from the system of equations.

$$\begin{cases} (x-x_1)^2+(y-y_1)^2+(z-z_1)^2=d_1^2 \\ (x-x_2)^2+(y-y_2)^2+(z-z_2)^2=d_2^2 \\ (x-x_3)^2+(y-y_3)^2+(z-z_3)^2=d_3^2 \end{cases}$$

Solving the system (analytical or iterative, e.g., by the least squares method) gives the absolute coordinates of the MR. Positioning accuracy in such systems reaches ±2 cm under optimal conditions.

Practical Recommendations for Implementation:

A. Synchronization: For accurate Time of Flight (ToF) measurement, it is recommended to use a radio channel to transmit the start time mark of the pulse, which eliminates clock synchronization error.
B. Temperature Compensation: The speed of sound depends on temperature; it is advisable to use a temperature sensor for dynamic correction.
C. Echo Filtering: Multipath reflections are possible in rooms; filtering by amplitude, arrival time, and matched signal processing is used.

Obstacle system

A set of ultrasonic rangefinders installed around the perimeter of the robot and touch sensors (bumpers) are used to detect obstacles. When an object is detected in the safety zone (less than 1 meter), the system initiates a bypass procedure. To increase reliability, it is possible to use sensors on multi-element photodetectors or computer vision cameras for obstacle classification (Shevchuk et al., 2024).

Structural Diagram

The structural diagram of the control device includes:

A. Microcontroller (central unit);
B. Radio communication module (for synchronization with beacons);
C. Motor drivers (H-bridges);
D. Sensor block (encoders, ultrasonic sensors, compass, temperature);
E. Indicator interfaces (LEDs, sound emitter).

CONTROL ALGORITHM

The control system operation algorithm implements a finite state machine with the following states:

Wait: Check system configuration, initialize sensors.

Route Planning: Calculate trajectory to the target point based on current coordinates.

Movement: Form control actions on engines to track the trajectory.

Obstacle Detection: Upon sensor trigger – stop, send sound signal, replan route.

Diagnostics: Control battery voltage, temperature, and motor electric current.

To ensure movement along a specified route, an algorithm minimizing deviation from the trajectory is used. The angular orientation of the robot is controlled using an electronic compass (magnetometer), which allows compensating for odometry drift (Vlasov et al., 2017).

SOFTWARE IMPLEMENTATION AND TESTING

The software is developed in C++ for the microcontroller platform. A modular structure is provided: equipment drivers, mathematics library (trilateration calculation, Kalman filters for data smoothing), decision logic.

System testing was carried out on a polygon with placed beacons. Main checked parameters:

A. Positioning accuracy (target value ±10 cm);
B. Reaction time to obstacle;
C. Motion stability under 100 kg load.

Test results showed that when using an ultrasonic system with temperature compensation, the coordinate determination error does not exceed 12% in dynamics, which is an acceptable result for navigation tasks in limited space. In static mode, positioning accuracy was ±8 cm. Comparison with SLAM methods showed the advantage of the proposed approach in conditions of smoke and absence of visual landmarks (Korsakov et al., 2024).

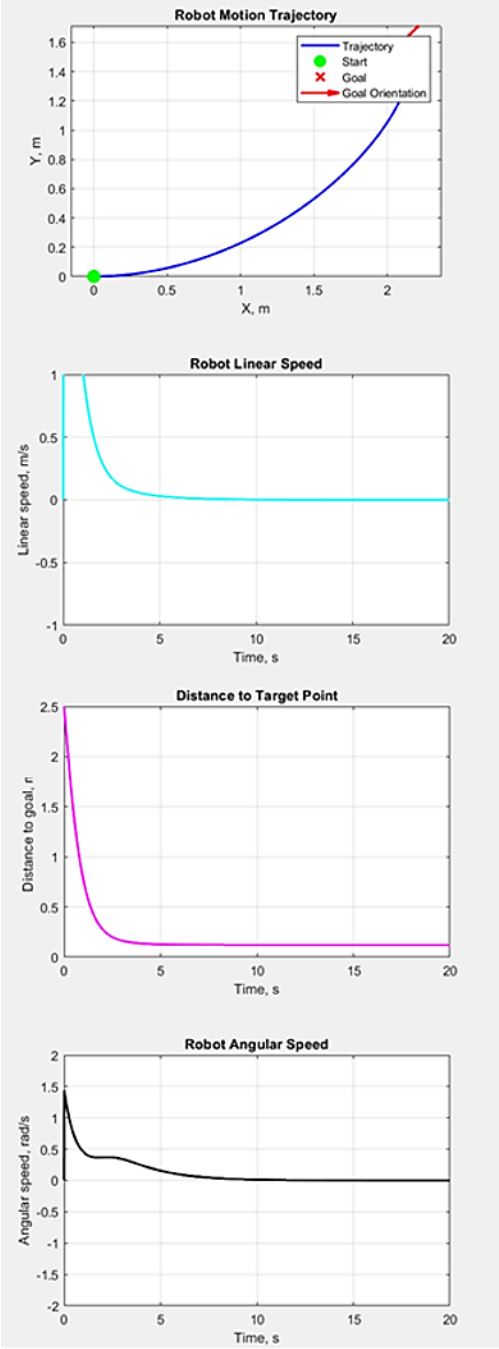


Fig. 1 Simulation of Mobile Robot Movement to a Given Point

CONCLUSIONS

During the work, a mobile robot control system structure was developed that meets the requirements of the technical assignment. The proposed combined navigation method (odometry + ultrasonic trilateration with temperature compensation) allows ensuring the necessary positioning accuracy with limited component costs. Implementation of obstacle avoidance algorithms and state control ensures safe operation of a 200 kg robot. Further research will be aimed at introducing SLAM methods for work in a dynamically changing environment and optimizing the system's energy consumption.

ACKNOWLEDGMENTS

This work was supported by KEGA project No. 001ŽU-4/2026: Creative environment for new 2nd and 3rd study degree programs in the field of mechanics and machine design.

REFERENCES

Ivanov D.A., Galieva T.G. (2022). Assessment of the Technical Condition of High-Voltage Insulators during Operation. Machines, 10(11), 1063.

Vlasov S.M., Boykov V.I., Bystrov S.V., Grigoryev V.V. (2017). Contactless means of local orientation of robots. St. Petersburg: ITMO University.

Galieva T.G., Sadykov M.F., Lyubishchev A.A. (2014). The Use of the Received Signal Strength Indicator (RSSI) as a Way to Register Partial Discharges. In: 2024 International Conference on Industrial Engineering, Applications and Manufacturing (ICIEAM), 173-178.

Safin M.A., Sadykov, R.D. (2023). Development of a mobile robot control system in Python programming language using Raspberry PI microcomputer. Automation. Modern Technologies, 77(12), 567-570.

Fukumura K., Kosaki, T. (2024). Fundamental Development of a Navigation Control System of a Wheelchair Robot with an Autonomous Mobile Robot. The Proceedings of Conference of Chugoku-Shikoku Branch, 62,(0), 09b3.

Vu T.V., Nguyen T.Q., Duc A.M.T. (2025). Implementation and Control System Development of a Differential Drive Wheeled Mobile Robot. Advances in Electrical and Electronic Engineering, 23(4).

Shevchuk O.S., Voytenko V.A., Vodichev V.A., Kalinin A.G. (2024). Development of an energy-efficient automatic control system of mobile industrial robot. Electrotechnical and Computer Systems, 39(115), 6-13.

Korsakov A., Ivanova V., Demcheva, A. et al. (2024). Development and Implementation of Neuromorphic Elements of the Information and Control System of a Mobile Robot. Optical Memory and Neural Networks, 33(S3), S504-S512.

Fedorov A.A., Orekhov S.Y., Veysman P.I., Vasilchuk, N.Y. (2021). Development of a control system for a mobile robot based on Markov processes. IOP Conference Series: Materials Science and Engineering, 1129(1), 012057.

Jhong B.G., Chen M.Y. (2023). Vector model-based robot-assisted control system for a wheeled mobile robot. Chung-kuo Kung Ch'eng Hsueh K'an, 46(5), 464-478.

Meshcheryakova A.A. (2024). Development of an automated control system for a mobile robot. Modeling of Systems and Processes, 17(1), 73-84.

Suarez-Rivera G., Muñoz-Ceballos N.D., Vásquez-Carvajal H.M. (2021). Development of an Adaptive Trajectory Tracking Control of Wheeled Mobile Robot. Revista Facultad de Ingeniería, 30(55), e12022.

Phuong C.T. (2023). Research and Development of Omnidirectional Mobile Robot Tracking Control Based on Artificial Intelligence. International Journal of Electronics and Communication Engineering, 10(3), 15-22.

Raikwar S., Fehrmann J., Herlitzius T. (2022). Navigation and control development for a four-wheel-steered mobile orchard robot using model-based design. Computers and Electronics in Agriculture, 202, 107410.

Lee L.W., Li I.H., Lu, L.Y. et al. (2022). Hardware Development and Safety Control Strategy Design for a Mobile Rehabilitation Robot. Applied Sciences, 12(12), 5979.

Shi Q. (2023). Trajectory tracking control method of mobile robot based on AVRX operating system. Journal of Computational Methods in Sciences and Engineering, 23(1), 253-265.

Tuleshov A.K., Seidakhmet A.Zh., Zemtsov, V.I. et al. (2025). Developing control in the ROS2 system for a mobile robot motor driver. Bulletin of the National Engineering Academy of the Republic of Kazakhstan, 97(3), 124-134.
Leiva M., Pérez-Zuñiga G., Cuellar F. (2025). Hybrid Control System for Navigation of an Articulated Mobile Tracked Robot. IFAC-PapersOnLine, 59(18), 361-366.
Bandyopadhyay S., Roy T. (2021). System Identification and Control of Wheeled Mobile Robot. IET Conference Proceedings, 2020(8), 170-176.
Nikitin Y.R., Turygin A.B., Stollmann V. (2022). Multilevel control of a transport robot. MM Science Journal, JUN, 5662–5669.
Božek P., Nikitin, Y. (2021). The development of an optimally-tuned PID control for the actuator of a transport robot. Actuators, 10(8).

Ivan Zajacko
University of Žilina, Faculty of Mechanical Engineering
Research and Service Center
Univerzitná 8215/1, 010 26 Žilina, **Slovakia**

Milan Saga
University of Žilina, Faculty of Mechanical Engineering
Research and Service Center
Univerzitná 8215/1, 010 26 Žilina, **Slovakia**

Zuzana Sagova
University of Žilina, Faculty of Mechanical Engineering
Research and Service Center
Univerzitná 8215/1, 010 26 Žilina, **Slovakia**

Pavol Bozek
University of Žilina, Faculty of Mechanical Engineering
Research and Service Center
Univerzitná 8215/1, 010 26 Žilina, **Slovakia**
e-mail: pavol.bozek@fstroj.uniza.sk

Appendix

Mobile Robot Movement Model to a Given Point in MATLAB

Below is a complete example of simulating a differential drive mobile robot moving to a specified target point. The code includes a kinematic model, control law, simulation, and trajectory visualization.

1. Main Simulation Script (main_robot_simulation.m)

```
%% Model of mobile robot movement to a given point
% Differential drive, position and orientation control
clear; clc; close all;

%% Robot Parameters
robot.L = 0.3;        % Robot base (distance between wheels), m
robot.R = 0.05;       % Wheel radius, m
robot.v_max = 1.0;    % Maximum linear speed, m/s
robot.w_max = 2.0;    % Maximum angular speed, rad/s

%% Control Parameters
Kp = 1.5;             % Proportional coefficient for distance
Ka = 2.0;             % Proportional coefficient for angle
Kb = 1.0;             % Orientation coefficient at target point

%% Initial Conditions
state = [0; 0; 0];    % [x; y; theta] - initial position and orientation
target = [2.0; 1.5];  % Target point [x_target; y_target]
target_theta = pi/4;  % Target orientation, rad

%% Simulation Parameters
dt = 0.01;            % Integration step, s
T = 20;               % Total simulation time, s
N = T/dt;             % Number of steps

%% Arrays for storing history
x_hist = zeros(1, N+1);
y_hist = zeros(1, N+1);
theta_hist = zeros(1, N+1);
v_hist = zeros(1, N+1);
w_hist = zeros(1, N+1);
x_hist(1) = state(1);
y_hist(1) = state(2);
theta_hist(1) = state(3);

%% Main simulation loop
for i = 1:N
    % Compute control actions
    [v, w] = compute_control(state, target, target_theta, Kp, Ka, Kb);

    % Limit speeds
    v = saturate(v, -robot.v_max, robot.v_max);
    w = saturate(w, -robot.w_max, robot.w_max);

    % Integrate kinematic model
    state = robot_kinematics(state, v, w, dt);

    % Save history
```

```
    x_hist(i+1) = state(1);
    y_hist(i+1) = state(2);
    theta_hist(i+1) = state(3);
    v_hist(i+1) = v;
    w_hist(i+1) = w;

    % Check goal achievement
    dist_to_goal = norm(state(1:2) - target);
    if dist_to_goal < 0.05 && abs(angle_diff(state(3), target_theta)) < 0.1
        fprintf('Goal reached at step %d, time = %.2f s\n', i, i*dt);
        x_hist = x_hist(1:i+1);
        y_hist = y_hist(1:i+1);
        theta_hist = theta_hist(1:i+1);
        v_hist = v_hist(1:i+1);
        w_hist = w_hist(1:i+1);
        break;
    end
end

%% Visualization of results
figure('Position', [100, 100, 1200, 800]);

% Motion trajectory
subplot(2, 2, 1);
plot(x_hist, y_hist, 'b-', 'LineWidth', 2);
hold on;
plot(0, 0, 'go', 'MarkerSize', 10, 'MarkerFaceColor', 'g'); % Start
plot(target(1), target(2), 'rx', 'MarkerSize', 15, 'LineWidth', 2); % Goal
quiver(target(1), target(2), cos(target_theta), sin(target_theta), ...
      0.3, 'r', 'LineWidth', 2); % Goal orientation
xlabel('X, m');
ylabel('Y, m');
title('Robot Motion Trajectory');
grid on;
axis equal;
legend('Trajectory', 'Start', 'Goal', 'Goal Orientation');

% Distance to goal
subplot(2, 2, 2);
dist_hist = sqrt((x_hist - target(1)).^2 + (y_hist - target(2)).^2);
plot((0:length(dist_hist)-1)*dt, dist_hist, 'm-', 'LineWidth', 2);
xlabel('Time, s');
ylabel('Distance to goal, m');
title('Distance to Target Point');
grid on;

% Linear speed
subplot(2, 2, 3);
plot((0:length(v_hist)-1)*dt, v_hist, 'c-', 'LineWidth', 2);
xlabel('Time, s');
ylabel('Linear speed, m/s');
title('Robot Linear Speed');
grid on;
ylim([-robot.v_max robot.v_max]);
```

```
% Angular speed
subplot(2, 2, 4);
plot((0:length(w_hist)-1)*dt, w_hist, 'k-', 'LineWidth', 2);
xlabel('Time, s');
ylabel('Angular speed, rad/s');
title('Robot Angular Speed');
grid on;
ylim([-robot.w_max robot.w_max]);

sgtitle('Simulation of Mobile Robot Movement to a Given Point');

%% Output final statistics
fprintf('\n=== Simulation Results ===\n');
fprintf('Initial position: [%.2f, %.2f, %.2f°]\n', ...
   x_hist(1), y_hist(1), rad2deg(theta_hist(1)));
fprintf('Final position: [%.2f, %.2f, %.2f°]\n', ...
   x_hist(end), y_hist(end), rad2deg(theta_hist(end)));
fprintf('Target position: [%.2f, %.2f, %.2f°]\n', ...
   target(1), target(2), rad2deg(target_theta));
fprintf('Total movement time: %.2f s\n', (length(x_hist)-1)*dt);
fprintf('Final distance to goal: %.4f m\n', dist_hist(end));
```

2. Kinematic Model Function (robot_kinematics.m)

```
function state_next = robot_kinematics(state, v, w, dt)
% Kinematic model of a differential drive mobile robot
% state = [x; y; theta] - current state
% v - linear speed, m/s
% w - angular speed, rad/s
% dt - integration step, s

x = state(1);
y = state(2);
theta = state(3);

% Kinematic equations
if abs(w) < 1e-6
   % Straight line motion
   x_next = x + v * cos(theta) * dt;
   y_next = y + v * sin(theta) * dt;
   theta_next = theta;
else
   % Arc motion
   x_next = x + (v/w) * (sin(theta + w*dt) - sin(theta));
   y_next = y + (v/w) * (cos(theta) - cos(theta + w*dt));
   theta_next = theta + w * dt;
end

% Angle normalization
theta_next = wrapToPi(theta_next);
state_next = [x_next; y_next; theta_next];
end
```

3. Control Computation Function (compute_control.m)

```
function [v, w] = compute_control(state, target, target_theta, Kp, Ka, Kb)
% Compute control actions for movement to goal
% Uses control law based on polar coordinates

x = state(1);
y = state(2);
theta = state(3);

% Distance to goal
dx = target(1) - x;
dy = target(2) - y;
rho = sqrt(dx^2 + dy^2);

% Angle to goal (in global coordinate system)
alpha = wrapToPi(atan2(dy, dx) - theta);

% Orientation error at target point
beta = wrapToPi(target_theta - atan2(dy, dx));

% Control law (polar regulator)
v = Kp * rho * cos(alpha);
w = Ka * alpha + Kb * beta;

% If robot is very close to goal, reduce speed
if rho < 0.1
   v = v * (rho / 0.1);
end
end
```

4. Helper Functions helper.m

```
function val = saturate(val, min_val, max_val)
% Limit value within specified bounds
if val < min_val
   val = min_val;
elseif val > max_val
   val = max_val;
end
end

function diff = angle_diff(angle1, angle2)
% Compute minimum difference between two angles
diff = wrapToPi(angle1 - angle2);
end
```

5. Extended Version with Obstacles (optional_robot_with_obstacles.m)

```
%% Model of movement with obstacle avoidance
clear; clc; close all;

%% Parameters
robot.L = 0.3;
robot.v_max = 1.0;
```

```
robot.w_max = 2.0;
dt = 0.01;
T = 30;

%% Obstacles (circular, [x, y, radius])
obstacles = [
   1.0, 0.8, 0.2;
   1.5, 1.2, 0.25;
   0.8, 1.5, 0.15
];

%% Initial and target conditions
state = [0; 0; 0];
target = [2.5; 2.0];
target_theta = pi/2;

%% Potential field for obstacle avoidance
Kp = 1.5;
Ka = 2.0;
K_obs = 5.0;       % Repulsion coefficient from obstacles
d_influence = 0.5;  % Influence zone of obstacles

%% Simulation
N = T/dt;
x_hist = zeros(1, N+1);
y_hist = zeros(1, N+1);
x_hist(1) = state(1);
y_hist(1) = state(2);

for i = 1:N
   % Basic control to goal
   [v_base, w_base] = compute_control(state, target, target_theta, Kp, Ka, 1.0);

   % Control from obstacles (potential field method)
   v_obs = 0;
   w_obs = 0;

   for j = 1:size(obstacles, 1)
      obs_x = obstacles(j, 1);
      obs_y = obstacles(j, 2);
      obs_r = obstacles(j, 3);
      dist = sqrt((state(1)-obs_x)^2 + (state(2)-obs_y)^2);

      if dist < d_influence + obs_r
         % Repulsion vector
         angle_to_obs = atan2(obs_y - state(2), obs_x - state(1));
         angle_diff = wrapToPi(angle_to_obs - state(3));

         % Repulsion force (increases when approaching)
         force = K_obs * (1/dist - 1/(d_influence + obs_r));
         v_obs = v_obs - force * cos(angle_diff);
         w_obs = w_obs - force * sin(angle_diff) * 2;
      end
   end
```

```
    % Total control
    v = saturate(v_base + v_obs, -robot.v_max, robot.v_max);
    w = saturate(w_base + w_obs, -robot.w_max, robot.w_max);

    % Integration
    state = robot_kinematics(state, v, w, dt);
    x_hist(i+1) = state(1);
    y_hist(i+1) = state(2);

    % Check goal achievement
    if norm(state(1:2) - target) < 0.05
        x_hist = x_hist(1:i+1);
        y_hist = y_hist(1:i+1);
        break;
    end
end

%% Visualization with obstacles
figure('Position', [100, 100, 800, 800]);
plot(x_hist, y_hist, 'b-', 'LineWidth', 2);
hold on;

% Obstacles
for j = 1:size(obstacles, 1)
    vis = circle_points(obstacles(j, 1), obstacles(j, 2), obstacles(j, 3));
    fill(vis(1,:), vis(2,:), 'r', 'FaceAlpha', 0.3, 'EdgeColor', 'r');
end

plot(0, 0, 'go', 'MarkerSize', 10, 'MarkerFaceColor', 'g');
plot(target(1), target(2), 'rx', 'MarkerSize', 15, 'LineWidth', 2);
xlabel('X, m');
ylabel('Y, m');
title('Robot Trajectory with Obstacle Avoidance');
grid on;
axis equal;
legend('Trajectory', 'Obstacles', 'Start', 'Goal');

%% Function for drawing circles
function vis = circle_points(cx, cy, r)
    theta = linspace(0, 2*pi, 100);
    vis(1,:) = cx + r * cos(theta);
    vis(2,:) = cy + r * sin(theta);
end
```

6. Results and Parameter Settings

Recommended Control Coefficients

Parameter	Range	Description
Kp	1.0-3.0	Speed of movement to goal
Ka	1.5-3.0	Turning speed to goal
Kb	0.5-2.0	Orientation accuracy at goal
K_obs	3.0-10.0	Repulsion force from obstacles

Typical Problems and Solutions

Oscillations near goal → Decrease Kp, add stop zone
Slow turning → Increase Ka
Insufficient orientation accuracy → Increase Kb
Collisions with obstacles → Increase K_obs or d_influence

7. Project Launch
% 1. Save all functions in separate .m files
% 2. In MATLAB, navigate to the folder with files
% 3. Execute the command:
main_robot_simulation

% 4. For the version with obstacles:
optional_robot_with_obstacles

This code provides a complete platform for modeling and debugging mobile robot navigation algorithms in MATLAB.