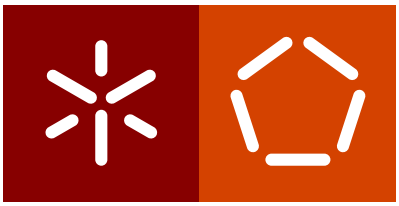


Universidade do Minho
Escola de Engenharia
Departamento de Engenharia Eletrónica
Nonlinear Dynamic Systems and Neural Networks

Autonomous Navigation: Report

Autonomous Navigation for mobile robots with modest computational resources

José Pires A50178
Mariana Duarte A85420

Supervised by:
Full Professor Estela Bicho

October 10, 2021

Contents

List of Figures

List of Listings

Symbols

Symbol	Description	Unit
a	linear acceleration of the robot	m/sˆ2
α	sigmoidal thresholded potential function	rad
β_1	maximum strenght of repulsion force	adim
β_2	decay rate of repulsion force with the distance increase	adim
C	sigmoidal sharpness constant	rad
c_{obs}	strenght of obstacles contribution to path velocity dynamics	adim
c_{tar}	strenght of target contribution to path velocity dynamics	adim
$c_{v,obs}$	relaxation rate for velocity dynamics when obstacles dominate	rad
$c_{v,tar}$	relaxation rate for velocity dynamics when target dominates	rad
d_i	sensed distance to the obstacle by sensor i	m
d_j	minimum distance for obstacle or target	m
d_{min}	minimum allowable distance to target	adim
Δt	Euler's timestep	s
$\Delta\theta$	angular distance between adjacent sensors (sensor sector)	rad
F_{obs}	overall obstacle avoidance dynamics	rad/s
f_{obs}	total repulsion force due to all obstacles contributions	rad/s
$f_{obs,i}$	repulsive force due to a virtual obstacle i	rad/s
f_{stoch}	Stochastic effect contribution to the planning dynamics	rad/s
f_{tar}	Target contribution to the planning dynamics	rad/s
k_v	linearity constant for linear velocity	adim
$\lambda_{obs,i}$	magnitude of the 'repulsion' force the virtual obstacle i exerts over the robot in heading direction dynamics	adim

Symbol	Description	Unit
λ_{tar}	magnitude of the 'attraction' force the target exerts over the robot in heading direction dynamics	adim
λ_v	magnitude of the 'attraction' force the target exerts over the robot in path velocity dynamics	adim
m	slope of the fixed point	adim
N	number of obstacle detection sensors	adim
ϕ	heading direction of the robot relative to an external reference axis	rad
ϕ^*	fixed point of the heading direction dynamics	rad
$\dot{\psi}_{max}$	maximal rate of fixed points' shift	rad/s
$\psi_{obs,i}$	angle by which each robot's sensor 'sees' an obstruction in respect to the external reference axis	rad
ψ_{tar}	angle by which the robot 'sees' the target in respect to the external world axes	rad
Q	magnitude of the stochastic force	adim
R_{robot}	robot's radius	m
R_{tar}	target's radius	m
σ_i	angular range over which each force-let exerts its repulsive effect	rad
$\tau_{obs,i}$	repulsion time for the repeller associated with the virtual obstacle i for obstacle avoidance in heading direction dynamics	s
τ_{tar}	relaxation time to the attractor for target acquisition in heading direction dynamics	s
τ_v	relaxation time to the attractor for target acquisition in path velocity dynamics	s
τ_{vdes}	time constant for the desired velocity increase in path velocity dynamics	s
θ_i	fixed angle each robot's sensor is mounted in respect to the external reference axis	rad

Symbol	Description	Unit
U	potential function of obstacle avoidance dynamics	rad
v	linear velocity of the robot	m/s
v_{des}	desired velocity	m/s
V_j	desired velocity for obstacle or target	m/s
v^*	fixed point of the path velocity dynamics	m/s
ε_n	Gaussian white noise of unit variance	adim
v_{max}	maximum linear velocity of the robot	m/s
x_{robot}	x coordinate of the robot's position	m
x_{tar}	x coordinate of the target position	m
y_{robot}	y coordinate of the robot's position	m
y_{tar}	y coordinate of the target position	m

1 Introduction

The present work, within the scope of the curricular unit of Nonlinear Dynamic Systems and Neural Networks, explores the problem of behavior generation for an autonomous mobile robot.

An autonomous mobile vehicle must be capable of pursuing a target while simultaneously avoiding collision with obstacles [1]. This desired behavior for the robot's motion can be described and structured using as behavioral variables (also known as state variables) the heading direction relative to an external world axis, ϕ, and the linear velocity of the robot, v. The temporal evolution of the behavioral variables is governed by nonlinear dynamics systems, described by ordinary differential equations [1]:

$$\frac{d\phi}{dt} = f(\phi,parameters), \tag{1.1}$$

$$\frac{dv}{dt} = g(v,parameters) \tag{1.2}$$

where the parameters depend, in general, of the sensorial information, hence the respective vectorial fields vary — number and position of fixed points and respective stability — as the robot moves or the environment changes (i.e., dynamic environments) [1].

The desired values for the behavioral variables — e.g., the direction the target — are made attractive force-lets (i.e., asymptotically stable states) of the dynamic system and unsidered values — e.g., the direction that obstacles are detected — are made repulsive force-lets (i.e., unstable states) of the dynamic system. The overall design idea for the autonomous navigation is to design such nonlinear dynamic systems such that it rests on, or very near of, an attractive force-let, thus yielding it robust to variations of the vectorial fields (robot motion and environment changes) and to noise.

Qualitative changes in the behavior of the robot — e.g., pass between obstacles or circumvent them — are generated by inducing proper bifurcations in the vectorial fields as a function of the parameters that depend on sensorial information.

The goal of the present work is to analyze, implement, simulate and tune the parameters of the dynamic systems yielding attractive and repulsive force-lets with adequate values, and that simultaneously the state variables follows closely one of the attractive force-lets as these vary. This is paramount to the design of

autonomous mobile robots as it contributes to the asymptotical stability of the system that generates the robot's behavior, yielding it robust to external disturbances.

1.1 Report organization

The present work is divided into small projects to understand and analyze the contribution of each component to the behavior of the robot. For each project the following tasks are performed:

1. Analytical study of the dynamic system
2. Implementation, in the simulator, of the dynamic system that generates the movement of the robot
3. Simulation and analysis of the behavior of the robot

This report is organized as follows:

- Chapter 1 lays out the theoretical foundations for this project and its overall goal.
- In Chapter 2 is studied, in isolation, the target acquisition behavior for the robot. Furthermore, it is analyzed the influence of the linearity of the vector field for the heading direction of the robot.
- In Chapter 3 is studied, in isolation, the obstacle avoidance behavior for the robot.
- In Chapter 4, the target acquisition and obstacles avoidance behaviors are integrated for fulfilling the overall goal of the robot — move to target in a collision free path — with the dynamic system for the target acquisition being nonlinear.
- In Chapter 5, the behaviors are also integrated for the same overall goal, but with the dynamic system for the target acquisition being linear.
- In Chapter 6, the control of the driving speed is added in a more systematic way to improve the overall dynamics.
- Chapter 7 gives a summary of the main findings of this work, defining the desired structure for the nonlinear dynamic systems of the behavioral variables — heading direction and linear velocity of the robot — and the range of acceptable values for the parameters and how to successfully implement and tune the system.

2 Target Acquisition

An autonomous mobile robot must be able to navigate to a target position. Information about the target position can be provided by a priori knowledge of the target coordinates — x_{tar}, y_{tar} — or the robot computer vision system may indicate the direction of the target (see Fig. 2.1).

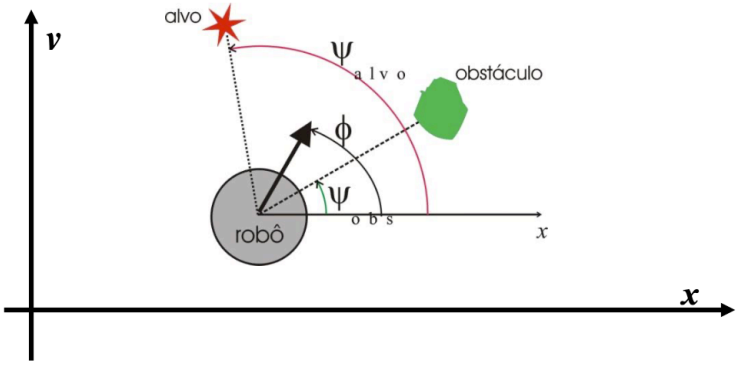


Figure 2.1: Target acquisition heading angle referencing to external world axes: The angle by which the robot 'sees', ψ_{tar}, the target in relation to external world axes may be yielded by the robot's computer vision system or may be computed if the target coordinates (x_{tar}, y_{tar}) are known and one has an estimative of the position of the robot (x_{robot}, y_{robot})

The target acquisition behavior can be yielded by dynamic systems for the heading direction and linear velocity:

$$\frac{d\phi}{dt} = f_{tar}(\phi,\psi_{tar}) + f_{stoch} \tag{2.1}$$

$$\frac{dv}{dt} = g_{tar}(v) \tag{2.2}$$

The term f_{tar} of the vectorial field in the equation (2.1) may have the nonlinear form:

$$f_{tar}(\phi,\psi_{tar}) = -\lambda_{tar}\sin(\phi - \psi_{tar}) \tag{2.3}$$

where λ_{tar} is a parameter that defines the magnitude of the 'attraction force' the target exerts over the robot and ψ_{tar} is the direction the robot 'sees' the target.

The term f_{stoch} of the vectorial field in the equation (2.1) represents the stochastic force that must be present to ensure escape from repellers within a limited time:

$$f_{stoch} = \sqrt{Q}\varepsilon_n \tag{2.4}$$

modelled as Gaussian white noise, ε_n, of unit variance, so that Q is the effective variance of the force [1].

2.1 Analytical study of nonlinear dynamic system defined by Eqs. (2.1) and (2.3)

In this section, an analytical study of the dynamic system defined by Eqs. (2.1) and (2.3) is performed. The fixed points of this dynamic system are determined and its stability is assessed, thus leading to successfull positioning of the attractor for the heading direction. Then, the phase portraits and bifurcation diagram are drawn, yielding the range of admissible values for the magnitude of the attractor.

2.1.1 Fixed points of the dynamic system

The fixed points of the dynamic system are the values, ϕ^*, for which the state of the system does not change, i.e., $f_{tar}(\phi^*) = 0$. Once placed at a fixed point, the system remains fixed in that state ($d\phi/dt = 0$), hence also named as constant solutions of the dynamic system. Graphically, the fixed points correspond to the intersection of the function $f(\phi)$ with the ϕ axis.

Fig. 2.2 illustrates the determination of the fixed points of the dynamic system, given by Eq. (2.5),yielding the general solution: $\phi^* = \psi_{tar} + k\pi$. As the heading direction $\phi \in [0, 2\pi[$ represents a circular phase space, the solutions can be limited to $\phi_1^* = \psi_{tar}, \phi_2^* = \psi_{tar} + \pi$.

$$\left.\frac{d\phi}{dt}\right|_{\phi=\phi^*} = f_{tar}(\phi^*) = 0 \tag{2.5}$$

2.1.2 Stability of the fixed points and relaxation time

The determination of the stability of fixed points is useful to understand its qualitative behavior. It can be determined analytically using Eq. (2.6), or infered graphically by observing the slope of $f_{tar}(\phi)$ at the fixed

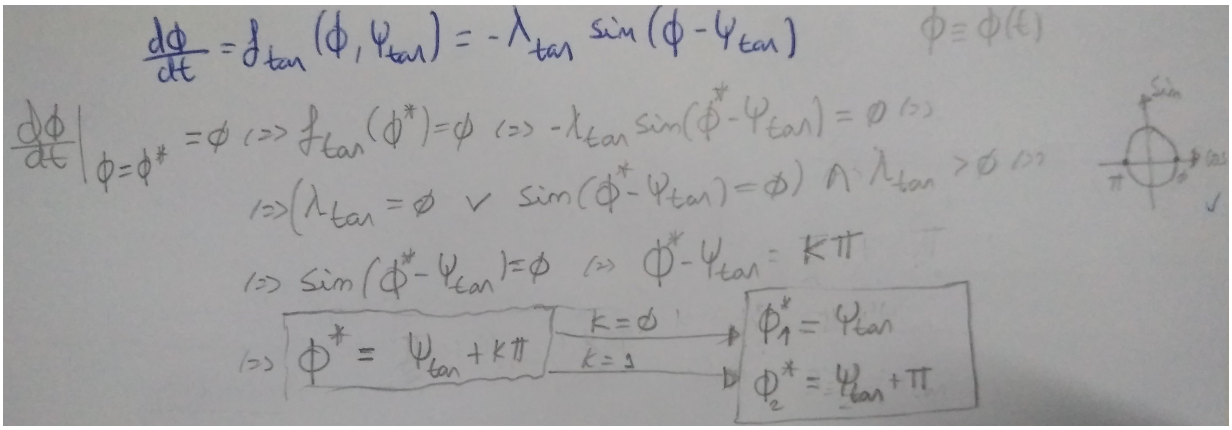


Figure 2.2: Target acquisition behavior: Determination of fixed points

points.

$$m = \left. \frac{df_{tar}(\phi)}{d\phi} \right|_{\phi=\phi^*} \tag{2.6}$$

Three cases arise, depending on the sign of m:

1. $m < 0$: ϕ^* is an asymptotically stable state, i.e., an attractor;
2. $m > 0$: ϕ^* is an unstable state, i.e., an repeller;
3. $m = 0$: nothing can be concluded about ϕ^* using the analytical method. The graphical method must be used.

The relaxation time, τ_{tar}, corresponds to the required time for the system state be significantly attracted to the asymptotically stable state when it is in the vicinity of this fixed point. It can be determined from the inverse of the slope on the vicinity of the fixed point, as given by Eq. (2.7):

$$\tau_{tar} = \frac{1}{|f'(\phi^*)|} \tag{2.7}$$

Fig. 2.3 illustrates the assessment of the stability of the fixed points and the relaxation time. It can be observed that the analytical method provides $m_1 = -\lambda_{tar}, m_2 = \lambda_{tar}$. Thus, for the attractor to be placed in the target direction, i.e. $\phi_1^* = \psi_{tar}$, and an repeller in the opposite direction, i.e., $\phi_2^* = \psi_{tar} + \pi$ as expected, $\lambda_{tar} > 0$. This way the robot can rapidly divert from the repeller and move to the attractor direction. The relaxation time is inversely proportional to the local time constant, i.e., inversely proportional to the magnitude of the attractive force-let $\tau_{tar} = 1/\lambda_{tar}$ s.

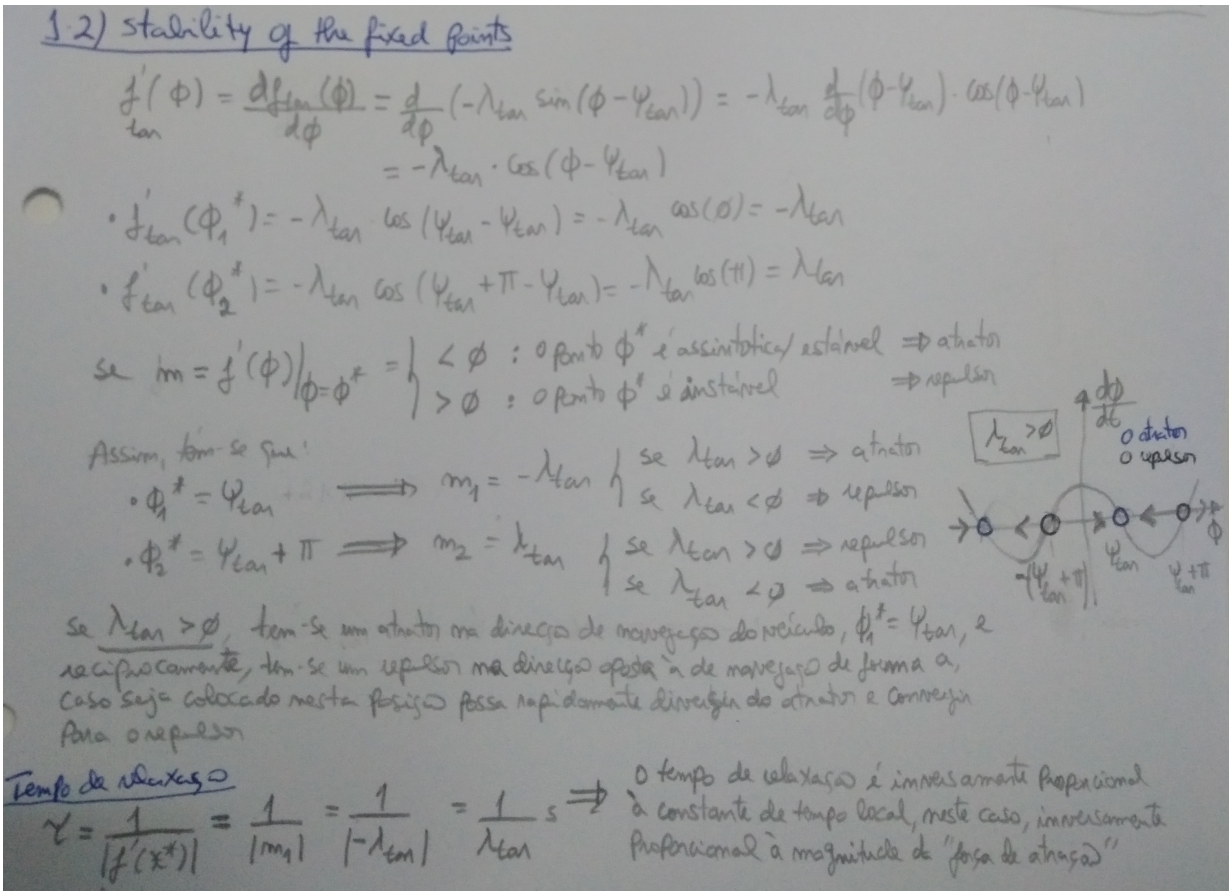


Figure 2.3: Target acquisition behavior: Stability of fixed points and relaxation time of the attractor

2.1.3 Phase portraits

The phase portrait of a dynamic system consists in a graphical representation that shows all its qualitatively different trajectories. The fixed points are represented as circles and the evolution direction of the state ϕ as the time progresses is indicated by arrows — convergent arrows to attractors and divergent arrows from repellers.

Fig. 2.4 depicts the possible phase portraits for the target acquisition dynamic system, depending on the sign of the parameter λ_{tar}:

- $\lambda_{tar} > 0$: an attractor is placed in the target direction and a repeller is placed in the opposite direction. This represents the intended behavior for the system.
- $\lambda_{tar} < 0$: a repeller is placed in the target direction and a attractor is placed in the opposite direction. This represents an undesired behavior for the system.

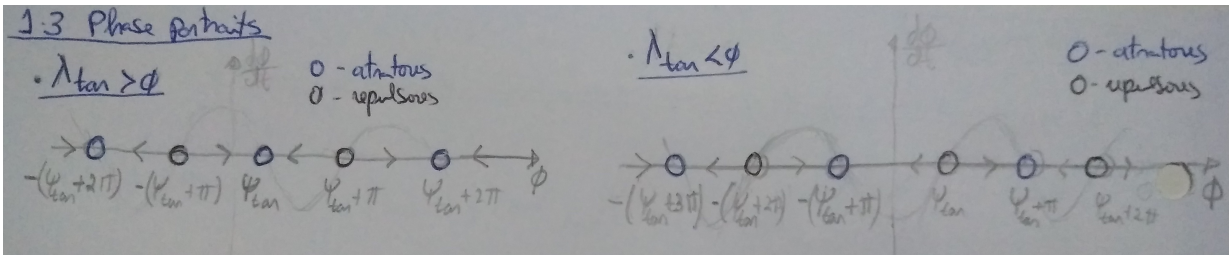


Figure 2.4: Target acquisition behavior: Phase portraits

2.1.4 Bifurcation diagram

Bifurcations arise as the qualitative behavior of the system changes, i.e., the number, nature and/or stability of the fixed points changes. A bifurcation point is the value of the parameter for which the qualitative change of the system behavior occurs. The bifurcation diagram consists in a graphical representation of the fixed points and respective stability as a function of a parameter of $f(\phi)$.

Fig. 2.5 depicts the bifurcation diagram for the target acquisition dynamic system, as a function of the parameter λ_{tar} for both fixed points – $\phi_1^* = \psi_{tar}, \phi_2^* = \psi_{tar} - \pi, \psi_{tar} > 0$:

- $\lambda_{tar} < 0$: the fixed point ϕ_1^* is unstable and ϕ_2^* is asymptotically stable.
- $\lambda_{tar} > 0$: the fixed point ϕ_1^* becomes asymptotically stable and ϕ_2^* is unstable.

$\lambda_{tar} = 0$ is a bifurcation point. This represents a transcritical bifurcation as there is an exchange in the stability between fixed points.

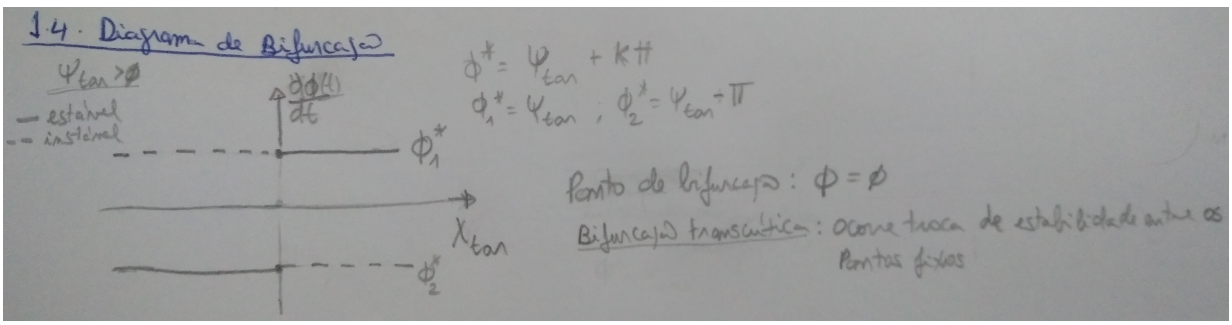


Figure 2.5: Target acquisition behavior: Bifurcation diagram

2.1.5 Range of values for λ_{tar}

For the target acquisition behavior, i.e. for the robot to move the target, an attractor must be placed in the direction of the target, thus $\phi_1^* = \psi_{tar}$ must be an asymptotically stable state of the system and

consequently $\lambda_{tar} > 0$.

2.2 Implementation of nonlinear dynamic system defined by Eqs. (2.1) and (2.3)

In this section is demonstrated the implementation of the nonlinear dynamic system for the heading direction. Additionally, a linear dynamic system for the linear velocity of the robot is also implemented.

The first step of the implementation is to convert the first order differential equation into an algebraic recursive equation by applying the forward Euler's method in a discrete form:

$$\phi(t + \Delta t) = \phi(t) + \Delta t f(\phi), \qquad \phi(t_0) = \phi_0 \tag{2.8}$$

where Δt is the Euler's step — the incremental timestep applied to the recursive equation. For smooth variation of the heading direction the Euler's step must be significantly smaller than the minimum time constant (in this case, the relaxation time), i.e.: $\Delta t \ll \tau_{tar}$. As a rule of thumb: $5\Delta t \leq \tau_{tar} \leq 10\Delta t$.

Next, the direction of the target, psi_{tar} must be determined, based on the known target coordinates and the robot's estimated coordinates with respect to the external reference axis, respectively, (x_{tar}, y_{tar}) and (x_{robot}, y_{robot}), taking into consideration the quadrant, as follows (see Fig. 2.1):

$$\psi_{tar} = atan2\left(\frac{y_{tar} - y_{robot}}{x_{tar} - x_{robot}}\right) \tag{2.9}$$

It is important to note that the target acquisition dynamics is dependent on the calibration of the heading direction of the robot in respect to the external reference axis to accurately locate it.

Then, both components of the dynamic system, f_{tar} and f_{stoch} can be computed using Eqs. (2.3) and (2.4) and added, yielding the complete dynamic system as given by Eq. (2.1). Next, the angular velocity of the vehicle can be determined, noting that $\omega_{robot} = d\phi/dt$, i.e., equal to the complete vector field.

Finally, linear velocity is defined, and, if desired a stop criterion for the distance to target. Then, the values of the angular and linear velocities of the robot can be passed as setpoints to the low-level code responsible for controlling these control variables.

Summarizing, the pseudocode for the target acquisition behavior is as follows:

1. Initialize robot: retrieve simulation timestep and robot characteristics

2. Set initial values (linear and angular velocities) and set robot's initial pose ($x_{robot}, y_{robot}, \phi_{robot}$)
3. Initialize target
4. While target <= targetNr
    a) Exchange information with the simulator
    b) Get vehicle's pose, target position and simulation timestep
    c) Trigger a simulation step
    d) Processing step
        i. Set parameters values: $\tau_{tar}, \lambda_{tar}, Q$
        ii. Compute ψ_{tar}
        iii. Compute f_{tar} and f_{stoch}
        iv. Compute resultant vector field f_{total} and assign it to angular velocity
        v. Set linear velocity
        vi. Define stop criterion for target distance (if desired)
    e) View dynamics: plot the target acquisition dynamics
    f) Set robot's angular and linear velocities
5. Terminate simulation and cleanup

As a result, the following generic Matlab code was implemented (Listing 2.1):

```
%% IMPORTANT:
% Before running this script, open the scenario in VREP, e.g
% Do not run simulation!
%% Last change: Estela Bicho Erlhagen, 18/10/2018
%% run this with scenario MobileRobotDyn_Tar.ttt

% Try to connect to simulator and initialize robot class:
vehicle = RobDyn_robot();

%get time step, dt, value (normally dt=50ms):
timestep = vehicle.get_simulation_timestep();

%Get robot physical characteristics:
[Rrobot,Theta_obs] = vehicle.get_RobotCharacteristics();
%Rrobot - robot radius (in cm)
```

```
%Theta_obs - Vector with angular position of each sensor i (i = 1, ..., 11) relative to the
%frontal direction (in rad)

% Set initial velocity vrobot = 0 (linear velocity) and wrobot = 0 (angular velocity)
% and get robot's initial pose
wrobot = 0.01; %rad/s
vrobot = 0; %cm/s
[xrobot, yrobot, phirobot] = vehicle.set_velocity(wrobot, vrobot);

% Define the (first) Target
ntarget=1; %initialize first target

%%%---------------------- Start Robot Motion Behavior -------------------
%%%-----------------------------------------------------------------------
while ntarget<=vehicle.TARGET_Number % until robot goes to target ntarget

   %% Robot interface
   % set and get information to/from vrep
   % avoid do processing in between ensure_all_data and trigger_simulation
   vehicle.ensure_all_data();

   % Get vehicle's pose (position in cm, phirobot in rad):
   [xrobot, yrobot, phirobot] = vehicle.get_vehicle_pose();

   %Get target position for selected Target (Target 1 or 2 - in cm):
   [XTARGET,YTARGET] = vehicle.get_TargetPosition(ntarget);

   %get simulation time
   sim_time = vehicle.get_simulation_time();

   %trigger simulation step
   vehicle.trigger_simulation();

   %% Processing step
   % Implement the dynamic systems that govern vehicle's behavior and
   % compute new angular and linear velocities...
   % The simulation timestep, dt, is stored in timestep (value is not changed
   % while simulation is running)
   % the simulation time is stored in sim_time.

   %%----------- BEGIN YOUR CODE HERE ----------- %
   %% 1. set parameter values
   % relation between timestep and euler step: tau_tar >> timestep
```

```
% magnitude of attraction, lambda_tar
% pre-factor of stochastic force, Q
dt = timestep;
tau_tar = 5 * dt;
lambda_tar = 1/tau_tar;
Q = 0.01;

%% 2. compute psi_tar
psi_tar = atan2(YTARGET - yrobot, XTARGET - xrobot);

%% 3. compute ftar
ftar = -lambda_tar * sin(phirobot - psi_tar);

%% 4. compute stochastic force, stoch
fstoch = sqrt(Q)* randn(1,1);

%% 5. Compute resultant vector field
f_total = ftar + fstoch;

%% 6. Compute angular velocity
wrobot = f_total;

%% 7. Set linear velocity
vrobot = 30; % cm/s

%% 8. Stop Criterion

%% view dynamics
% visualization of the dynamics (target acquisition)
phi_plot = (0:10:360)*pi/180;
lphi = length(phi_plot);
ftar_plot = -lambda_tar * sin(phi_plot - psi_tar);
fstoch_plot = sqrt(Q) * randn(lphi, 1);
ftotal_plot = ftar_plot;% + fstoch_plot;
figure(1);
clf;
plot(phi_plot * 180/pi, ftotal_plot);
yl = ylim; % get current axis y-axis limits
hold on
phi_robot_plot = phirobot;
if(phirobot < 0)
    phi_robot_plot = phirobot + 2*pi;
end
```

```
    line([ phi_robot_plot * 180/pi, phi_robot_plot * 180/pi], [yl(1), yl(2)]);
103 %plot(phirobot, f_total);
    hold off

    %%------------ END OF YOUR CODE ------------

    % set robot angular velocity (rad/s) and robot linear velocity (cm/s)
109 vehicle.set_velocity(wrobot, vrobot);
    %--------------------------------------------------------------------------
111 end
    % cleanup
113 vehicle.terminate();
```

Listing 2.1: Generic implementation of target acquisition behavior (not tuned)

Several scenarios have been simulated in CoppeliaSim/Matlab — scenario **MobileRobotDyn_tar.ttt** — for different linear velocity conditions namely:

1. Robot with rotational motion only: v = 0 m/s;
2. Robot moving at constant speed: v = K m/s;
3. Robot moving at a velocity defined by a vectorial field;

2.2.1 Rotational motion only

In this first scenario, one considers only the existence of the rotational motion of the robot in respect to its center of mass, i.e., v = 0 m/s. The parameters were tuned to guide the robot for the desired target direction.

The scenario **MobileRobotDyn_tar.ttt** was loaded in CoppeliaSim and the robot was placed in the opposite direction of the target (worst case scenario), as illustrated in Fig. 2.6. Then, the initial values of angular and linear velocities were set and the parameters were tuned as indicated in Listing 2.2:

- $v_{robot} = 0$: robot does not exhibit translational motion.
- $\tau_{tar} = 5\Delta t$: minimum value to ensure forward Euler's method convergence.
- $Q = 0.01$: sufficient effective variance of the Gaussian white noise to enable qualitative differences in the behavior (turn left/right).

```
% Set initial angular and linear velocities of the robot and get its initial pose
wrobot = 0.01; %rad/s
vrobot = 0; %cm/s
[xrobot, yrobot, phirobot] = vehicle.set_velocity(wrobot, vrobot);

   %% 1. set parameter values
   dt = timestep;
   tau_tar = 5 * dt;
   lambda_tar = 1/tau_tar;
   Q = 0.01;
   %% 7. Set linear velocity
   vrobot = 0; % cm/s
```

Listing 2.2: Target acquisition behavior: Parameter tuning for v = 0 m/s

Fig. 2.7 illustrates the target acquisition dynamics in the final condition. Initially, the heading direction, ϕ, sits in a repeller $\phi = 50 + 180°$ (positive slope). The heading direction diverts from this fixed point as it represents an unstable state and tries to redirect to the attractor (turning in clockwise direction) located at $\phi = 50°$ (negative slope). When it reaches this asymptotically stable state the robot remains there, as intended. This validates the behavior of orientation to the target.

Additionally, it is important to note the effect of the stochastic force: in some occasions the robot follows a different trajectory for target orientation by turning in counter-clockwise direction, depending on the randomized value of the Gaussian white noise, ε_n.

2.2.2 Robot moving at constant speed: v = K

After validating the behavior of orientation to the target, it is time to implement the actual movement to the target by making the linear velocity $v \neq 0$ m/s. First off, the linear velocity was set to a constant value and the parameters were tuned, as illustrated in Listing 2.3, where the only relevant change in respect to Section 2.2.1 is the value of the linear velocity.

Fig. 2.8 depicts the simulation performed for $v = 30$ cm/s. The orientation behavior remains the same, but now the robot moves to the desired target location (see also Video ./videos/tar-2-2.mp4), eventually colliding with it at that speed, as no stop criterion is defined and the linear velocity is independent from the distance to the target. Increasing the linear velocity of the vehicle increases the turning radius, slowing down the orientation to the target. If the velocity is set too high and the vehicle is close enough to the arena walls, it may crash against it before even turning (or while turning).

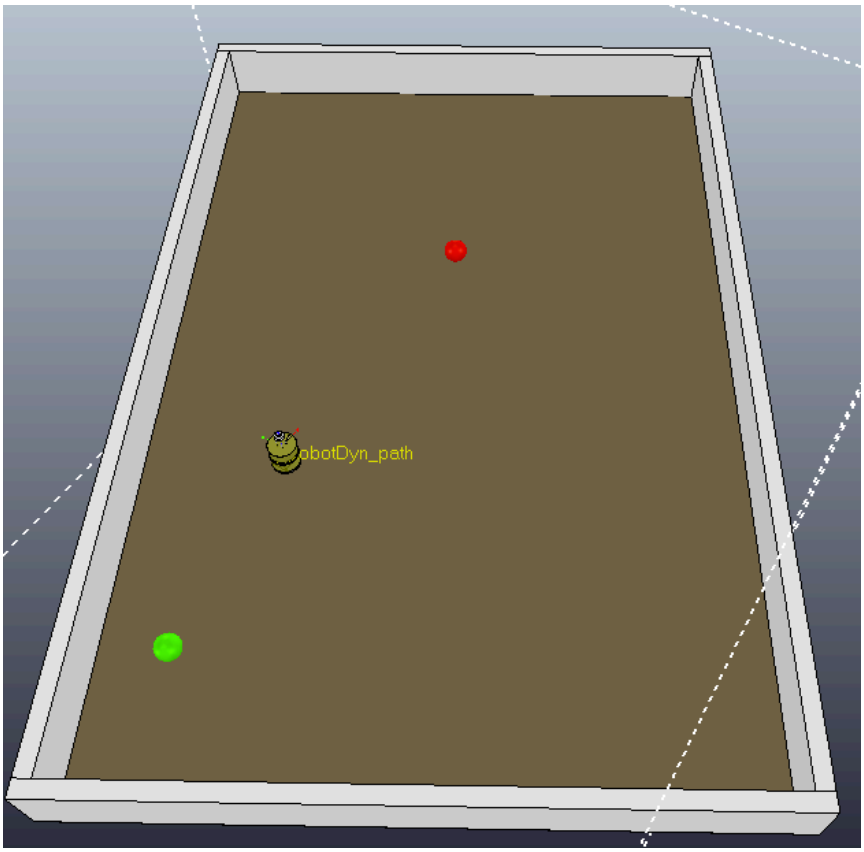


Figure 2.6: Target acquisition behavior: Simulation in CoppeliaSim for v = 0 m/s

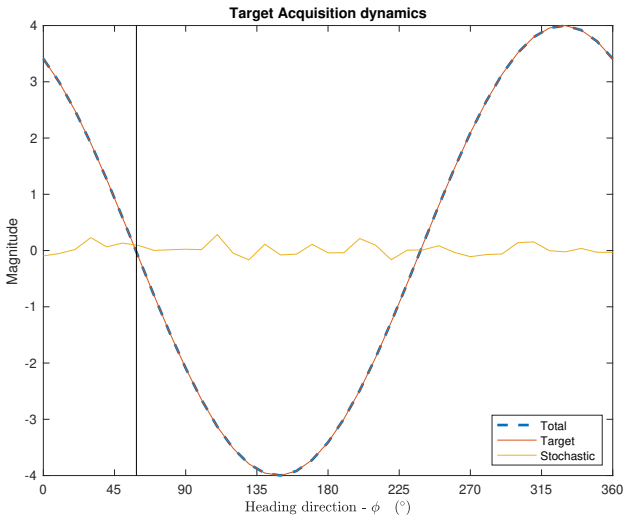


Figure 2.7: Target acquisition behavior: dynamics for v = 0 m/s

The dynamics remains identical to the one in Section 2.2.1 (see Fig. 2.7), although slightly slower due to the increased turning radius for target orientation.

```
%% 1. set parameter values
dt = timestep;
tau_tar = 5 * dt;
lambda_tar = 1/tau_tar;
Q = 0.01;
%% 7. Set linear velocity
vrobot = 30; % cm/s
```

Listing 2.3: Target acquisition behavior: Parameter tuning for v = 30 cm/s

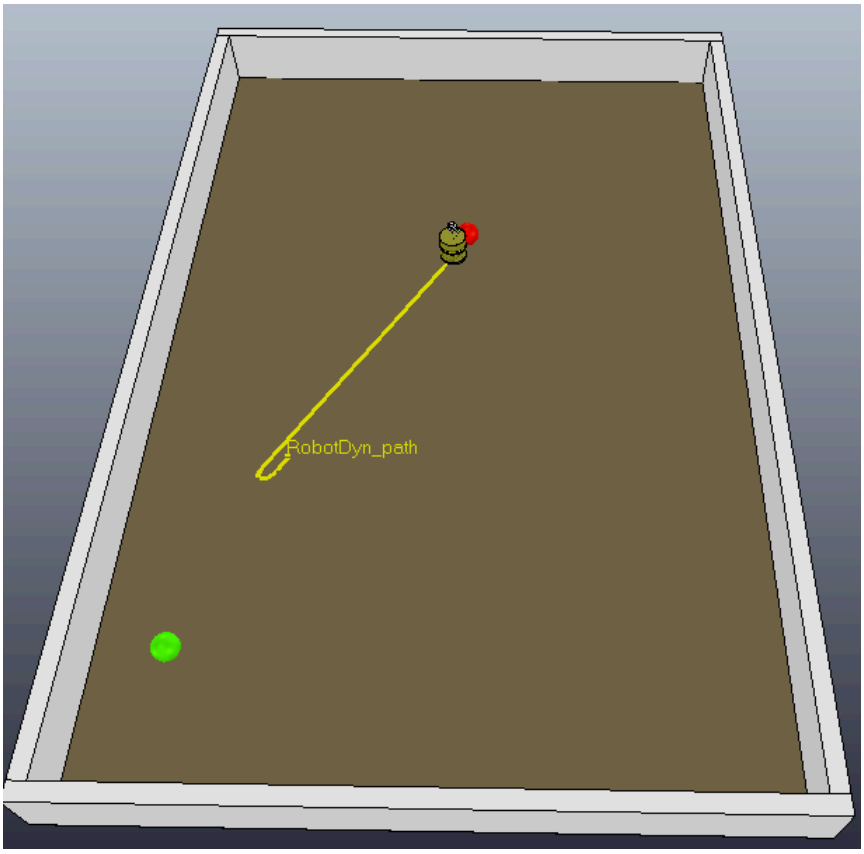


Figure 2.8: Target acquisition behavior: Simulation in CoppeliaSim for v = 30 cm/s

2.2.3 Robot moving at a velocity defined by a vectorial field

Previously, the orientation and movement to the target were validated. However, the linear velocity of the vehicle was constant and independent of the distance to the target, which can be problematic especially if the velocity is high or a stop criterion is undefined.

In this section a linear vectorial field is defined for the linear velocity of the robot as given by Eq. (2.10):

$$\frac{dv}{dt} = g_{tar}(v) = \lambda_v(v - v_{des}) \tag{2.10}$$

where v_{des} is the desired velocity and depends on the distance to the target.

First, an analytical study of the vectorial field was performed for the determination of its fixed points and respective stability — similar to the one in Section 2.1 — and the phase portraits and bifurcation diagram were plotted (see Fig. 2.9). There is a fixed point at the desired velocity, i.e., $v^* = v_{des}$, which is asymptotically stable (an attractor) if and only if $\lambda_v < 0$, yielding the intended behavior. Additionally, the relaxation time to the attractor is given by Eq. (2.7), yielding $\tau_v = 1/|\lambda_v|$.

The next step is to convert the first order differential equation into a algebraic recursive equation in a discrete form (Eq. (2.11)):

$$v(t + \Delta t) = v(t) + \Delta t g_{tar}(v), \qquad v(t_0) = v_0 \tag{2.11}$$

Noting that $g_{tar}(v) = dv(t)/dt = a(t)$, one can write:

$$v(t + \Delta t) = v(t) + \Delta t a(t), \qquad v(t_0) = v_0 \tag{2.12}$$

which represents the equation of the uniformly varied rectilinear motion, with $a(t)$ being the acceleration of the robot. If $a(t) > 0$ the motion is accelerated and reciprocally if $a(t) < 0$ the motion is retarded. Thus, one can analyse the sign of $a(t)$ to understand the motion of the robot. Recalling that $\lambda_v < 0$:

$$a(t) = g(v) = \begin{cases} > 0, & (v - v_{des}) \leq 0 \\ < 0, & (v - v_{des}) > 0 \end{cases} \leftrightarrow a(t) = g(v) = \begin{cases} > 0, & v \leq v_{des} \\ < 0, & v > v_{des} \end{cases} \tag{2.13}$$

the motion will be accelerated if $v \leq v_{des}$ — which represents the initial condition, ramping from rest to the desired velocity — and retarded if $v > v_{des}$ — the inertia and delayed actuation (as a consequence of the timestep) may lead to excessive velocity. On the other hand, the linear velocity should converge to a constant value in the vicinity of the desired velocity ($a(t) = 0 \rightarrow v(t + \Delta t) = v(t)$).

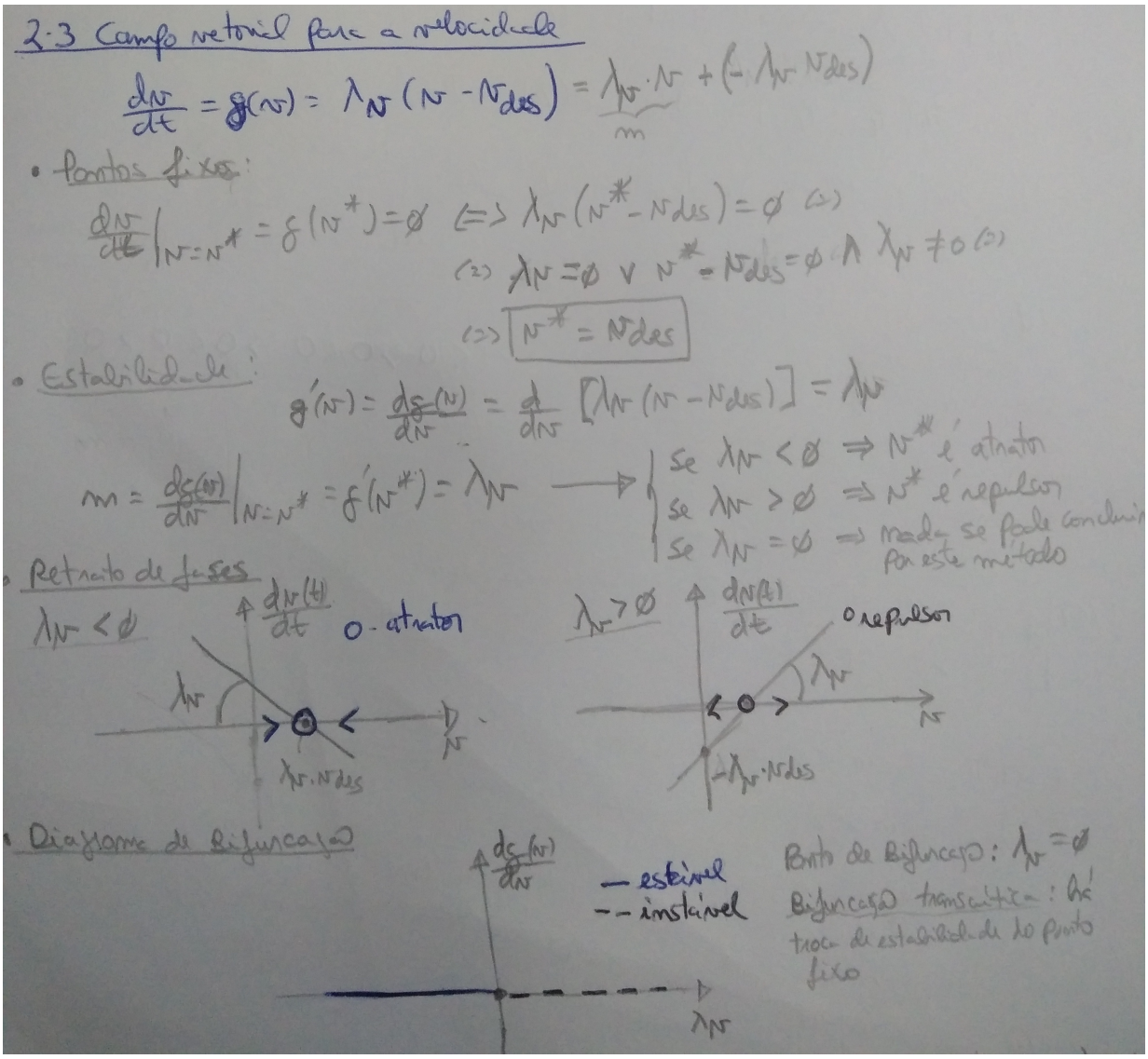


Figure 2.9: Target acquisition behavior: determination of the velocity dynamics

With this in mind, one must determine the desired velocity as a function of the distance to the target. Ideally, one desires a steady increase of velocity with the distance and a plateau for the maximum velocity, which can be expressed as:

$$v_{des}(d) = v_{max} - v_{max}e^{-\frac{d}{\tau_{vdes}}} \quad \leftrightarrow \quad v_{des}(d) = v_{max}\left(1 - e^{-\frac{d}{\tau_{vdes}}}\right) \tag{2.14}$$

where v_{max} is the maximum velocity (cm/s) and τ_{vdes} is the time constant for the desired velocity increase.

Fig. 2.10 illustrates a comparison of functions for the desired velocity as a function of distance, $v_{des}(d)$.

The maximum velocity considered was 80 cm/s. The dashed line corresponds to the linear function $v_{des}(d) = k_v d$, with $k = 1$, which acts as a guideline, although it could also be used, provided the v_{max} threshold was defined. For $\tau_{vdes} \leq 20$ s, the robot would hit the target fast, which can be problematic due to path corners and inertia; for $\tau_{vdes} \geq 75$ s it can be slow. A tradeoff between velocity and path smoothness must be considered, as detailed further ahead.

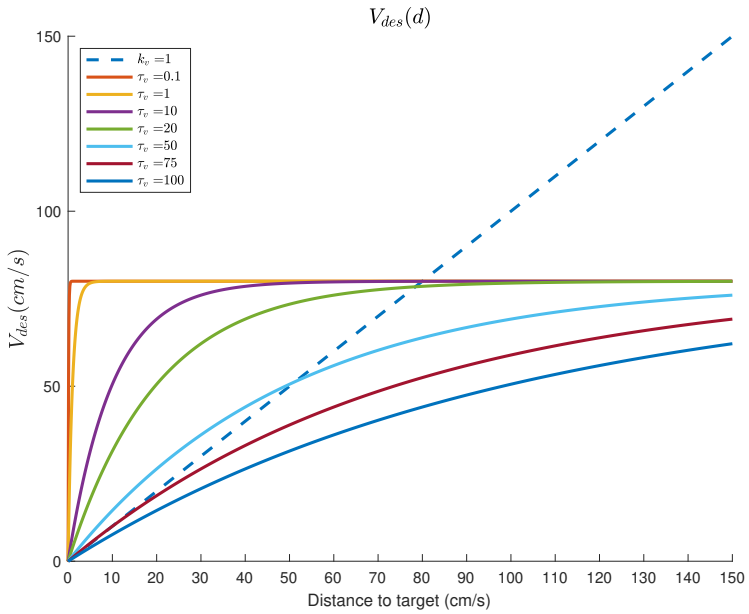


Figure 2.10: Target acquisition behavior: comparison of functions $v_{des}(d)$

Lastly, the distance to the target, $glsd$, must be determined. It can be seen from Fig. 2.1 the distance to the target is given by the euclidean distance in the cartesian plane, i.e.:

$$d = \sqrt{(x_{robot} - x_{tar})^2 + (y_{robot} - y_{tar})^2} \tag{2.15}$$

However, the coordinates of the robot and the target are of their centers of mass, which indicates an horizontal shift of magnitude d_{min} (minimum distance) should be performed to the exponential functions illustrated in Fig. 2.10, yielding:

$$v_{des} = \begin{cases} v_{max}\left(1 - e^{-\frac{d-d_{min}}{\tau_{vdes}}}\right), & d \geq d_{min} \\ 0, & d < d_{min} \end{cases} \tag{2.16}$$

as illustrated in Fig. 2.11. The minimum distance is calculated as $d_{min} = R_{robot} + R_{tar}$, where R_{robot} and

R_{tar} are the radius of the robot's base (22.5 cm) and target (20 cm), respectively, yielding $d_{min} = 42.5$ cm.

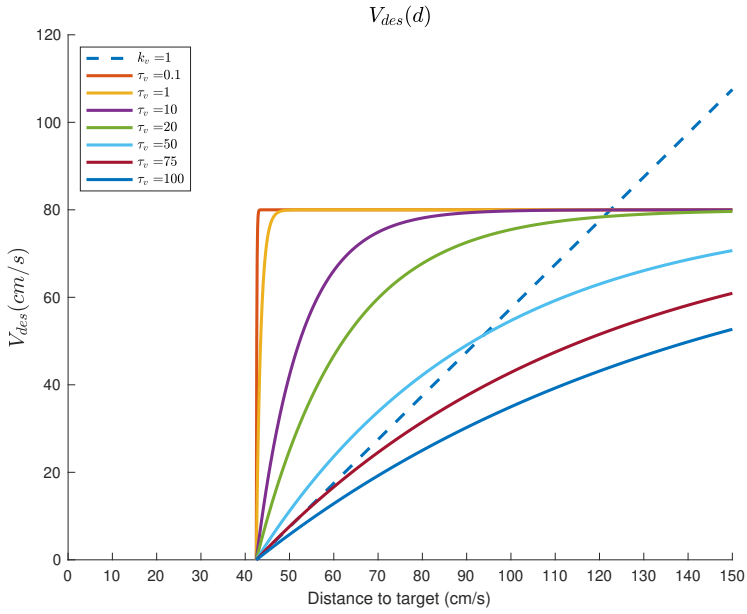


Figure 2.11: Target acquisition behavior: comparison of functions $v_{des}(d)$ (with minimum distance shift)

The vector field for linear velocity was implemented (see (Listing 2.4)) and the parameters were tuned. The desired behavior for the robot is faster orientation to target, i.e., heading direction dynamics takes precedence over velocity dynamics. Additionally, the $v_{des}(d)$ must have a slower influence on the velocity dynamics (smoother curve) to avoid steep accelerations. Thus, the hierarchy of time constants for the system is: $\tau_{tar} \ll \tau_v \ll \tau_{vdes}$.

```
%%----------- BEGIN YOUR CODE HERE ----------- %
%% 1. set parameter values
dt = timestep;
% phi
tau_tar = 10 * dt;
lambda_tar = 1/tau_tar;
% velocities
tau_v = 5*tau_tar;
lambda_v = - 1/tau_v;
Q = 0.01; % factor of stochastic force
          %k_v = 1/5;
vrobot_max = 80; % cm/s
tau_vdes = 150*tau_v; %1500 * tau_tar;
Rtar = 20; % target radius
```

```
d_min = Rrobot + Rtar + 5; % minimum distance to target

%% 2. compute psi_tar
psi_tar = atan2(YTARGET - yrobot, XTARGET - xrobot);

%% 3. compute ftar
ftar = -lambda_tar * sin(phirobot - psi_tar);

%% 4. compute stochastic force, stoch
fstoch = sqrt(Q)* randn(1,1);

%% 5. Compute resultant vector field
f_total = ftar + fstoch;

%% 6. Compute angular velocity
wrobot = f_total;

%% 7. Set linear velocity
dist_tar = sqrt( (YTARGET - yrobot)^2 + (XTARGET - xrobot)^2 )
if(dist_tar < d_min)
    v_des = 0;
else
    %v_des = k_v * abs( (dist_tar - d_min) ) % linear
    v_des = vrobot_max * (1 - exp(- (dist_tar - d_min) / tau_vdes) )
end

g_v = lambda_v * (vrobot - v_des)
vrobot = vrobot + dt * g_v; % cm/s
```

Listing 2.4: Implementation of target acquisition behavior with dynamic field for linear velocity

The heading direction and velocity dynamics were also plotted, alongside with $v_{des}(d)$ (see Listing 2.5) to aid the behavior analysis.

```
%% V_des(d)
f = figure(2);
f.Name = 'V_des(d)';
f.ToolBar = 'none'; f.MenuBar = 'figure';
clf;
%
ax = gca;
ax.YLim = [0, vrobot_max + 20];
ax.XLim = [0, 600];
```

```
yl = ax.YLim; % get current axis y-axis limits
xl = ax.XLim; % get current axis y-axis limits
hold on

d = d_min:0.1:600;
vdes_plot = vrobot_max .* (1 - exp(- (d - d_min) / tau_vdes) );
p = plot(d, vdes_plot, 'LineWidth', 2, 'LineStyle', '-');
l = line([dist_tar, dist_tar], [yl(1), yl(2)]);
l.Color = 'black';
l = line([xl(1), xl(2)], [v_des, v_des]);
l.Color = 'black';
ax.Title.String = '$V_{des} (d) $';
ax.Title.FontSize = 14;
ax.Title.Interpreter = 'latex';
ax.XLabel.String = 'Distance to target (cm/s)';
ax.YLabel.String = '$V_{des} (cm/s) $';
ax.YLabel.Interpreter = 'latex';
ax.YLabel.FontSize = 14;
hold off

%% g(v)
f = figure(3);
f.Name = 'g(v)';
f.ToolBar = 'none'; f.MenuBar = 'figure';
clf;
hold on

vrobot_plot = 0:0.1:vrobot_max;
g_v_plot = lambda_v * (vrobot_plot - v_des);
p = plot(vrobot_plot, g_v_plot, 'LineWidth', 2, 'LineStyle', '-');

ax = gca;
yl = ax.YLim; % get current axis y-axis limits
xl = ax.XLim; % get current axis y-axis limits
l = line([vrobot, vrobot], [yl(1), yl(2)]);
l.Color = 'black';
l = line([xl(1), xl(2)], [0, 0]);
l.LineStyle = '--';
l.Color = 'black';
ax.Title.String = 'Linear velocity dynamics: g(v)';
ax.Title.FontSize = 12;
ax.XLabel.String = 'Velocity (cm/s)';
ax.YLabel.String = 'Magnitude';
```

```
53 hold off
```

Listing 2.5: Implementation of target acquisition behavior with dynamic field for linear velocity

Fig. 2.12 illustrates the simulation in CoppeliaSim for the linear velocity dynamics implementation (see also Video ./videos/tar-2-3-linear-vel.mp4). It can be seen that the robot successfully moves to target, remaining at a minimum distance. The turning radius is small, as the heading dynamics has a stronger effect over the robot than the velocity dynamics. Additionally, during this simulation it was noted the robot accelerates in the beginning to $1/3$ of the overall distance $g_{tar}(v) > 0$ and decelerates until it reaches the target (within the minimum distance). The final state of the simulation is illustrated for $v_{des}(d)$ and $g_{tar}(v)$ in Figs. 2.13 and 2.14. It can be seen that the desired velocity reaches 0 m/s and the corresponding dynamics $g_{tar}(v)$ is near the attractor in deceleration, as expected.

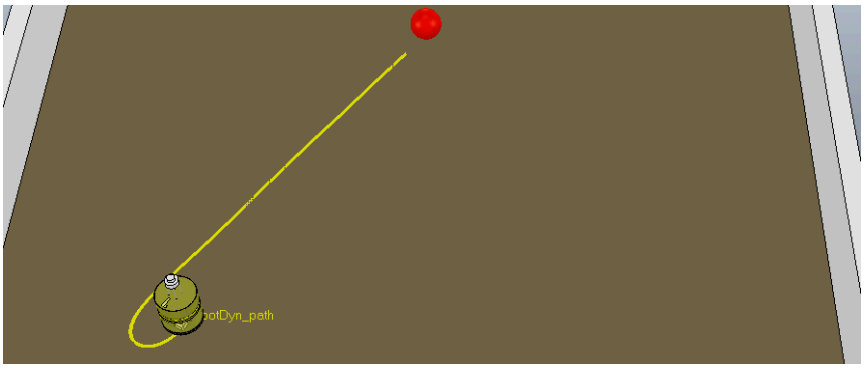


Figure 2.12: Target acquisition behavior: velocity dynamics simulation in CoppeliaSim

2.3 Discussion

As aforementioned (See Section 2.2.3), the robot moves to the target with variable velocity: it accelerates in the beginning to $1/3$ of the overall distance $g_{tar}(v) > 0$ and decelerates until it reaches the target (within the minimum distance). The heading direction dynamics takes precedence over the velocity dynamics to allow smooth navigation to the target: the robot orientates itself faster to the target than it actually moves. Additionally, the $v_{des}(d)$ has a slower influence on the velocity dynamics (smoother curve) to avoid steep accelerations. Thus, the hierarchy of time constants for the system is: $\tau_{tar} \ll \tau_v \ll \tau_{vdes}$. The heading direction dynamics is illustrated in Fig. 2.7 and in Figs. 2.13 and 2.14 can be seen that the desired velocity reaches 0 m/s and the corresponding dynamics $g_{tar}(v)$ is near the attractor in deceleration, as expected.

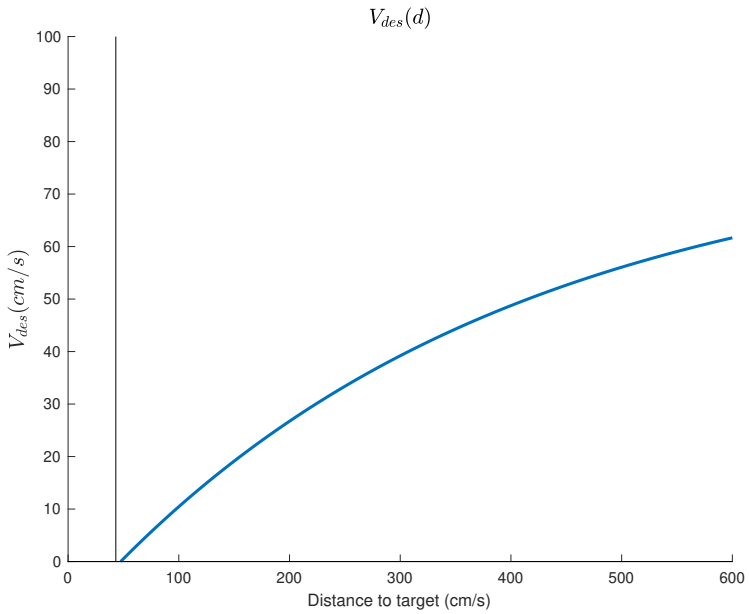


Figure 2.13: Target acquisition behavior: velocity dynamics simulation – $v_{des}(d)$

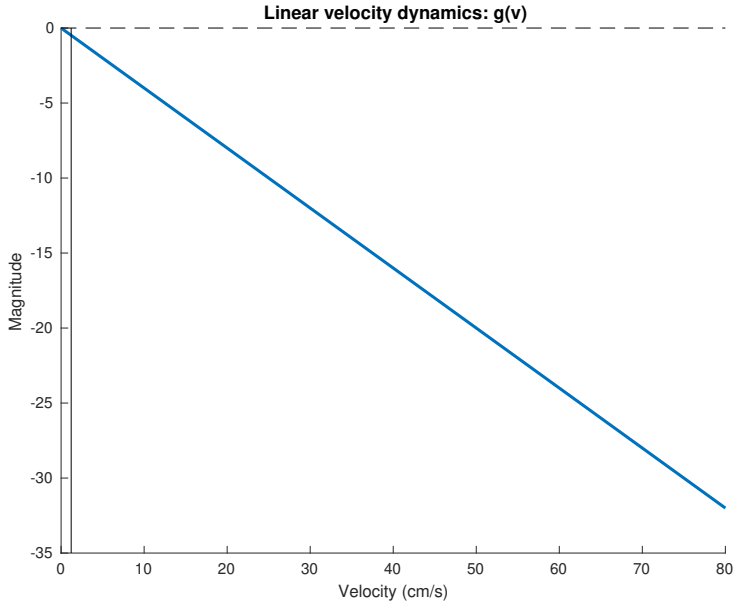


Figure 2.14: Target acquisition behavior: velocity dynamics simulation – $g_{tar}(v)$

2.4 Linear dynamic system for heading direction

In this section a linear dynamic system for the heading direction was analyzed:

$$\frac{d\phi}{dt} = f_{tar}(\phi) = -\lambda_{tar}(\phi - \psi_{tar}) \tag{2.17}$$

The fixed points and its stability was determined. The phase portraits were plotted and compared to the nonlinear counterpart. Next, the linear dynamic system was implemented and simulated, observing some differences of the navigation paths due to higher angular velocity range. Lastly, the linear and nonlinear dynamic system were compared, determing the best suited for the task of target acquisition behavior of the robot.

2.4.1 Fixed points

The fixed points of the linear dynamic system can be determined as its constant solutions, i.e.:

$$\begin{aligned} \tfrac{d\phi}{dt}\big|_{\phi=\phi^*} = f_{tar}(\phi^*) = 0 \quad &\leftrightarrow \quad -\lambda_{tar}(\phi^* - \psi_{tar}) = 0 \\ \leftrightarrow \quad (\lambda_{tar} = 0 \vee \phi^* - \psi_{tar} = 0) \wedge \lambda_{tar} \neq 0 \quad &\leftrightarrow \quad \boxed{\phi^* = \psi_{tar}} \end{aligned} \tag{2.18}$$

There is one fixed point at $\phi^* = \psi_{tar}$.

2.4.2 Stability

The determination of the stability of fixed points is useful to understand its qualitative behavior. It can be determined analytically using Eq. (2.19):

$$m = \left.\frac{df_{tar}(\phi)}{d\phi}\right|_{\phi=\phi^*} = f'_{tar}(\phi^*) \quad \leftrightarrow \quad \boxed{m = -\lambda_{tar}} \tag{2.19}$$

Analyzing the possible cases for the slope at the fixed points, it can be observed that:

$$m = \begin{cases} < 0, & \lambda_{tar} > 0 \quad \rightarrow \quad \text{attractor} \\ > 0, & \lambda_{tar} < 0 \quad \rightarrow \quad \text{repeller} \\ = 0, & \lambda_{tar} = 0 \quad \rightarrow \quad \text{inconclusive} \end{cases} \tag{2.20}$$

The desired heading direction dynamics requires that an attractor is placed in the target direction, i.e.

$\phi^* = \psi_{tar}$ is an attractor, thus $\lambda_{tar} > 0$. The relaxation time for the attractor is given by:

$$\tau_{tar} = \frac{1}{|f'_{tar}(\phi^*)|} \quad \leftrightarrow \quad \boxed{\tau_{tar} = \frac{1}{\lambda_{tar}}} \tag{2.21}$$

2.4.3 Phase portraits

Fig. 2.15 illustrates the different phase portraits for the linear dynamic system of the heading direction $\phi \in [0, 2\pi[$ (circular phase space). Comparing the phase portraits between linear and nonlinear dynamic systems, respectively Figs. 2.15 and 2.4, it can be observed that the former has one fixed point only — attractor or repeller depending on the value of the parameter λ_{tar}, but not both on the same phase portrait — while the latter has two fixed points — one attractor and one repeller on each phase portrait.

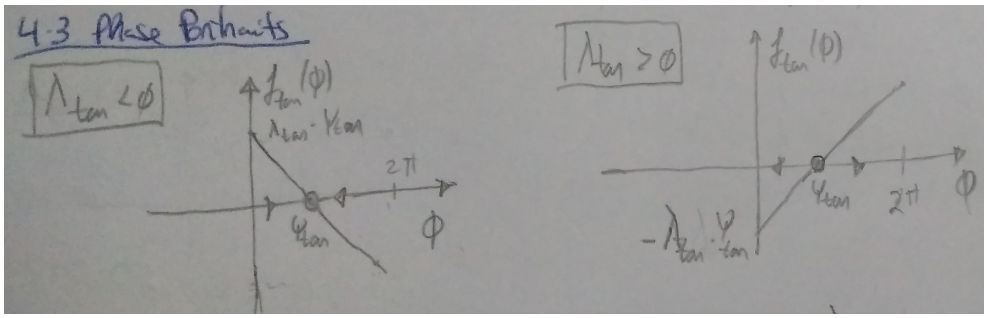


Figure 2.15: Target acquisition behavior: linear dynamic system for heading direction — phase portraits

2.4.4 Implementation

The implementation of the linear dynamic system for the heading direction was performed considering also the linear dynamic system for linear velocity, i.e., modifying the implementation from Section 2.2.3 with respect to the heading direction dynamics as follows:

```
% dynamic system
ftar = -lambda_tar * (phirobot - psi_tar);
% plot
ftar_plot = -lambda_tar * (phi_plot - psi_tar);
```

Listing 2.6: Implementation of linear dynamic system for heading direction

The same considerations, in respect to the nonlinear dynamic system, needed to be applied, namely $\Delta t \ll \tau_{tar}$.

2.4.5 Discussion

Fig. 2.16 illustrates the simulation of the linear dynamic system for heading direction implemented (see also ./videos/tar-4-4.mp4), and Fig. 2.17 contains the final state of the dynamics. Comparing Fig. 2.16 with Fig. 2.12, respectively the simulation of linear and nonlinear dynamics, it can be observed the smaller curvature radius corresponding to the initial orientation to the target for the former. This is due to the higher magnitude of the attractive force-let (linear) (higher angular velocity) for linear dynamics, as compared to the repulsive and attractive force-lets (sinusoidal) for nonlinear dynamics. The nonlinear dynamic system is better suited for target acquisition behavior of the robot, as it provides smoother paths — narrower angular velocity range.

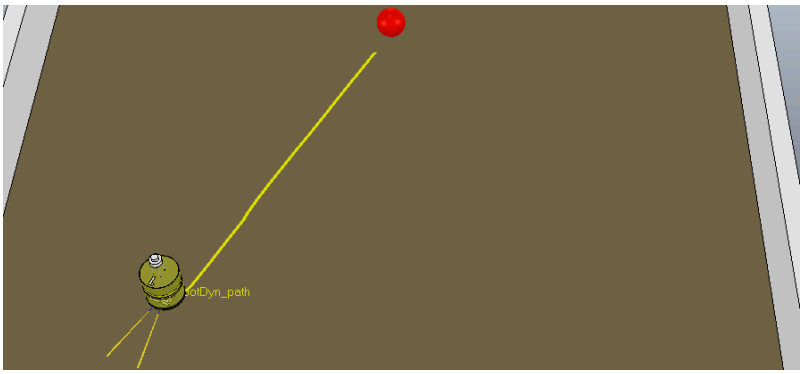


Figure 2.16: Target acquisition behavior: linear dynamic system for heading direction — simulation

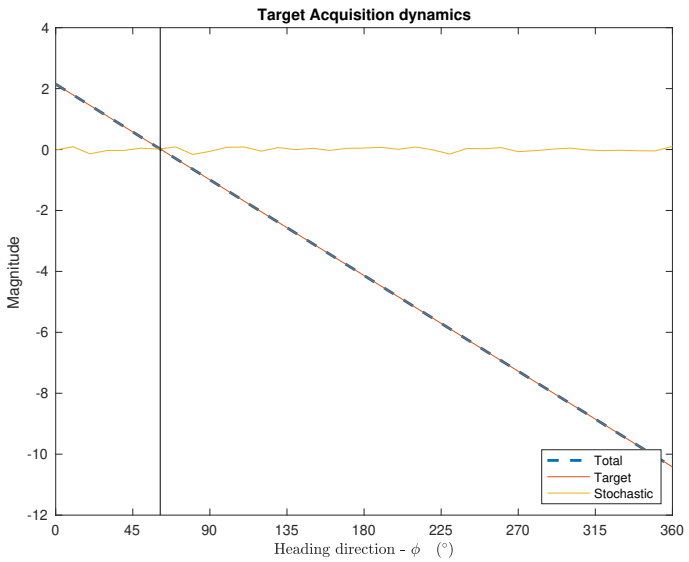


Figure 2.17: Target acquisition behavior: linear dynamic system for heading direction — dynamics (final state)

3 Obstacle Avoidance

While moving to a target, a robot must also avoid obstacles that may appear — obstacle avoidance. To avoid obstacles, the robot must firstly detect them, in this case, using infrared radiation sensors mounted in a circular support which is centered in respect to the rotation axis of the robot (Fig. 3.1). Each sensor is mounted in a fixed direction θ_i in respect to frontal direction of the robot, i.e., the heading direction. Thus, each sensor points into a direction $\psi_{obs,i} = \phi + \theta_i$, in respect to the external reference axis [1].

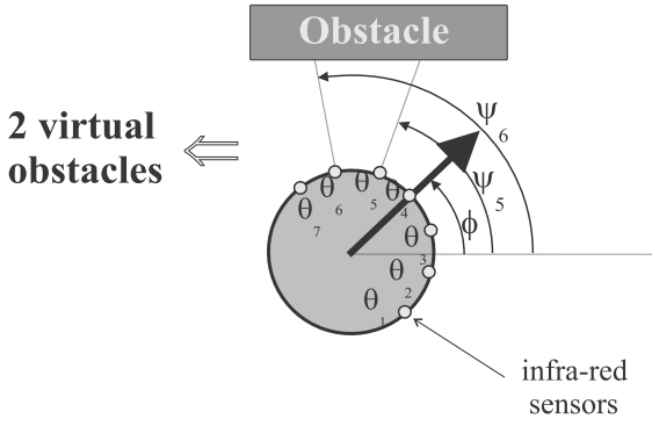


Figure 3.1: Obstacle avoidance behavior: sensor placement

The strategy adopted consists in assuming that each sensor i specifies a virtual obstacle in the direction $\psi_{obs,i}$ if an obstruction is detected in that direction. Each virtual obstacle is modelled by a repulsive force centered in the direction the respective sensor points out:

$$f_{obs,i}(\phi) = \lambda_{obs,i}(\phi - \psi_{obs,i}) \exp\left(-\frac{(\phi - \psi_{obs,i})^2}{2\sigma_i^2}\right) = \lambda_{obs,i}(-\theta_i) \exp\left(-\frac{(-\theta_i)^2}{2\sigma_i^2}\right) \quad (3.1)$$

It must be pointed out that the only diference $\phi - \psi_{obs,i} = -\theta_i$, which is fixed and known, goes into the heading direction dynamics, thus yielding the calibration of the system in respect to the external reference axis irrelevant.

The strength of the repulsion, $\lambda_{obs,i}$, from the virtual obstacle at direction $\psi_{obs,i}$, is a decreasing function

of the sensed distance, d_i — high for small distances and low for high distances and null for $d_i \geq d_{max}$ (sensor range maximum) — which can be exponential:

$$\lambda_{obs,i} = \beta_1 \exp\left(-\frac{d_i}{\beta_2}\right) \tag{3.2}$$

where β_1 determines the maximum strenght of repulsion and β_2 is the decay rate of the repulsion force with the distance increase.

The angular range over which the force-let exerts its repulsive effect, σ_i is given by:

$$\sigma_i = \arctan\left(\tan\left(\frac{\Delta\theta}{2}\right) + \frac{R_{robot}}{R_{robot} + d_i}\right) \tag{3.3}$$

where $\Delta\theta$ is the angular distance between adjacent sensors (sensor sector), R_{robot} is the radius of the robot, and d_i is the distance estimated by the sensor i.

The contributions from all the sensors are summed, thus, yielding the overall obstacle avoidance dynamics as follows:

$$\frac{d\phi}{dt} = F_{obs}(\phi) = f_{obs}(\phi) + f_{stoch} \tag{3.4}$$

where N is the number of sensors, in this case, $N = 11$, f_{obs} is the total repulsion force due to all obstacles contributions (Eq. (3.5)), and f_{stoch} is the stochastic force given by Eq. (2.4).

$$f_{obs}(\phi) = \sum_{i=1}^{N} f_{obs,i}(\phi) \tag{3.5}$$

3.1 Analytical study

In this section the analytical study of the overall obstacle avoidance dynamics (Eq. (3.4)) is performed. The fixed points are determined and its stability assessed. The phase portraits and bifurcation diagram are plotted. Lastly, the range of admissible values for the parameter β_1 and the repulsion time constant are computed.

3.1.1 Fixed points

The fixed points for the dynamic system given by Eq. (3.4) are computed, considering that only one obstruction is detected, i.e.:

$$\frac{d\phi}{dt} = F_{obs}(\phi) = f_{obs,1}(\phi) \tag{3.6}$$

The fixed points are the constant solutions of the dynamic system, i.e.:

$$\begin{aligned} &\tfrac{d\phi}{dt}\big|_{\phi=\phi^*} = f_{obs,1}(\phi^*) = 0 \quad \leftrightarrow \quad \lambda_{obs,1}(\phi^* - \psi_{obs,1}) \exp\left(-\tfrac{(\phi^*-\psi_{obs,1})^2}{2\sigma_i^2}\right) = 0 \quad \leftrightarrow \\ &\leftrightarrow \quad (\lambda_{obs,1} = 0 \vee \phi^* - \psi_{obs,1} = 0 \vee \exp\left(-\tfrac{(\phi^*-\psi_{obs,1})^2}{2\sigma_i^2}\right) = 0) \wedge \lambda_{obs,i} \neq 0 \\ &\leftrightarrow \quad \boxed{\phi^* = \psi_{obs,1}} \end{aligned} \tag{3.7}$$

There is one fixed point at $\phi^* = \psi_{obs,1}$.

3.1.2 Stability

The determination of the stability of fixed points is useful to understand its qualitative behavior. It can be determined analytically evaluating the slope in the vicinity of the fixed point. First, let one compute the derivative:

$$\begin{aligned} &f'_{obs,1}(\phi) = \tfrac{df_{obs,1}(\phi)}{d\phi} = \tfrac{d}{d\phi}\left(\lambda_{obs,1}(\phi - \psi_{obs,1}) \exp\left(-\tfrac{(\phi-\psi_{obs,1})^2}{2\sigma_i^2}\right)\right) \quad = \\ &= \tfrac{d}{d\phi}\Big(\phi - \psi_{obs,1}\Big)\lambda_{obs,1} \exp\left(-\tfrac{(\phi-\psi_{obs,1})^2}{2\sigma_i^2}\right) + \tfrac{d}{d\phi}\left(\exp\left(-\tfrac{(\phi-\psi_{obs,1})^2}{2\sigma_i^2}\right)\right)\lambda_{obs,1}(\phi^* - \psi_{obs,1}) \leftrightarrow \\ &\leftrightarrow \quad f'_{obs,1}(\phi) = \lambda_{obs,1} \exp\left(-\tfrac{(\phi-\psi_{obs,1})^2}{2\sigma_i^2}\right)\left(1 - \tfrac{(\phi-\psi_{obs,1})^2}{\sigma_1^2}\right) \end{aligned} \tag{3.8}$$

Then, computing the derivative at the fixed point yields:

$$\begin{aligned} &f'_{obs,1}(\phi^*) = f'_{obs,1}(\psi_{obs,1}) = \lambda_{obs,1} \exp\left(-\tfrac{(\psi_{obs,1}-\psi_{obs,1})^2}{2\sigma_i^2}\right)\left(1 - \tfrac{(\psi_{obs,1}-\psi_{obs,1})^2}{\sigma_1^2}\right) \\ &\leftrightarrow \quad \boxed{f'_{obs,1}(\phi^*) = \lambda_{obs,1}} \end{aligned} \tag{3.9}$$

Analyzing the possible cases for the slope at the fixed point, it can be observed that:

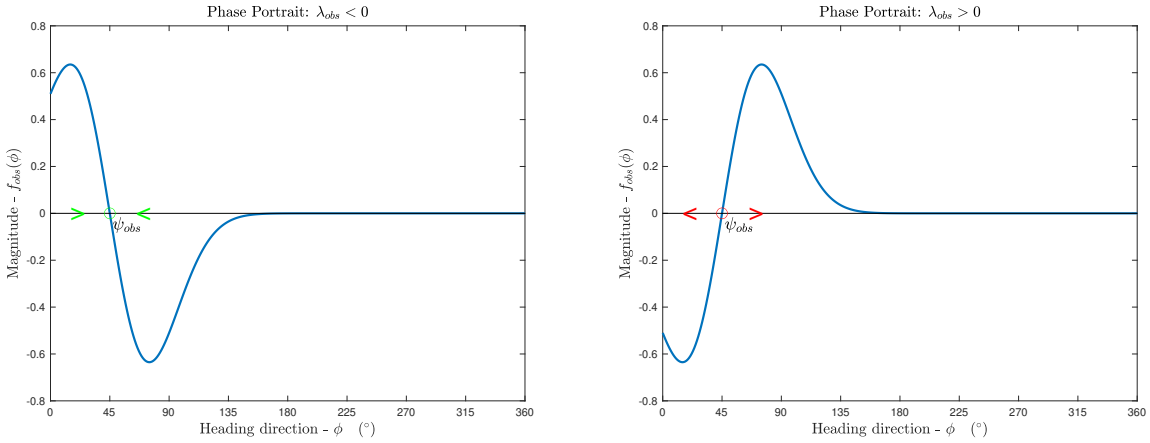
$$m = \begin{cases} < 0, & \lambda_{obs} < 0 \quad \rightarrow \quad \text{attractor} \\ > 0, & \lambda_{obs} > 0 \quad \rightarrow \quad \text{repeller} \\ = 0, & \lambda_{obs} = 0 \quad \rightarrow \quad \text{inconclusive} \end{cases} \tag{3.10}$$

The desired heading direction dynamics requires that a repeller is placed in the obstacle direction, i.e. $\phi^* = \psi_{obs,1}$ is an repeller, thus $\lambda_{obs} > 0$. The repulsion time for the repeller is given by:

$$\tau_{obs,1} = \frac{1}{|f'_{obs,1}(\phi^*)|} \quad \leftrightarrow \quad \boxed{\tau_{obs,1} = \frac{1}{\lambda_{obs,1}}} \tag{3.11}$$

3.1.3 Phase portraits

Fig. 3.2 illustrates the phase portraits for the dynamic system responsible for the obstacle avoidance behavior. It can be observed that for $\lambda_{obs} < 0$, $\phi^* = \psi_{obs}$ is an attractor and conversely if $\lambda_{obs} < 0$, $\phi^* = \psi_{obs}$ is an repeller. As previously mentioned the latter represents the desired behavior.



(a) $\lambda_{obs} < 0$: attractor

(b) $\lambda_{obs} > 0$: repeller

Figure 3.2: Obstacle avoidance behavior: Phase portraits

3.1.4 Bifurcation diagram

Fig. 3.3 depicts the bifurcation diagram for the target acquisition dynamic system, as a function of the parameter λ_{obs} ($\psi_{tar} \in [0, 2\pi[$):

- $\lambda_{obs} < 0$: the fixed point ϕ^* is asymptotically stable.
- $\lambda_{obs} > 0$: the fixed point ϕ^* is unstable.

$\lambda_{obs} = 0$ is a bifurcation point. This represents a transcritical bifurcation as there is an exchange in the stability in the fixed point.

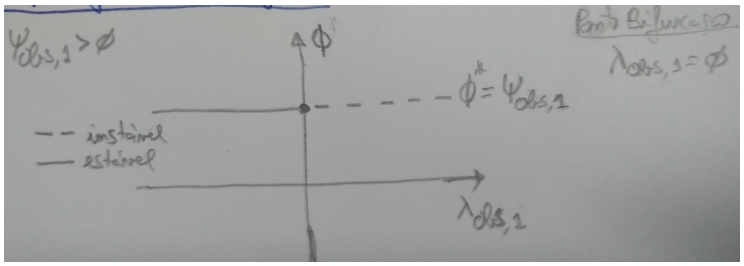


Figure 3.3: Obstacle behavior: Bifurcation diagram

3.1.5 Range of values for β_1

The parameter β_1 represents the maximum strength of the repulsive forcelet, given by:

$$\beta_1 = \max(\lambda_{obs,i}) = \frac{1}{\min(\tau_{obs,i})} \tag{3.12}$$

i.e., is the inverse of the minimum time constant, $\tau_{obs,i}$, between all sensors. The minimum time constant must be much greater than the Euler's step, Δt, and considering reasonable boundaries, one has:

$$5\Delta t < min(\tau_{obs}) < 10\Delta t \quad \leftrightarrow \quad \frac{1}{10\Delta t} < \beta_1 < \frac{1}{5\Delta t} \tag{3.13}$$

3.1.6 Repulsion time constant

This was previously calculated (see Section 3.1.2).

3.2 Implementation

In this section, the implementation of the nonlinear dynamic system for obstacle avoidance and simulation in CoppeliaSim using scenario **MobileRobotDyn_Tar_Obs.ttt** is described for several different environment scenarios. This aided the comprehension of the influence of several parameters and enabled its successfull tuning, yielding the desired obstacle avoidance behavior for the robot.

The first step of the implementation is to convert the first order differential equation into an algebraic recursive equation by applying the forward Euler's method in a discrete form:

$$\phi(t + \Delta t) = \phi(t) + \Delta t F_{obs}(\phi), \qquad \phi(t_0) = \phi_0 \tag{3.14}$$

where Δt is the Euler's step — the incremental timestep applied to the recursive equation and $F_{obs}(\phi)$ is given by Eq. (3.4). Simplying, both components of the dynamic system, f_{obs} and f_{stoch} can be computed separately using Eqs. (3.5) and (2.4) and added, yielding the complete dynamic system as given by Eq. (3.4). Next, the angular velocity of the vehicle can be determined, noting that $\omega_{robot} = d\phi/dt$, i.e., equal to the complete vector field.

For smooth variation of the heading direction the Euler's step must be significantly smaller than the minimum time constant (in this case, the repulsion time), i.e., $\Delta t \ll \min(\tau_{obs})$, as previously determined.

Next, for each sensor are computed the contribution to the repulsive force-let, namely the direction the obstacle is seen, $\psi_{obs,i}$, magnitude of repulsion, $\lambda_{obs,i}$, the range of repulsion, σ_i, and the effective contribution to the repulsive forcelet, $f_{obs,i}$. The latter is accumulated, yielding f_{obs}.

Finally, linear velocity is defined, and, if desired a stop criterion for the distance to target. Then, the values of the angular and linear velocities of the robot can be passed as setpoints to the low-level code responsible for controlling these control variables.

Summarizing, the pseudocode for the obstacle avoidance behavior is as follows:

1. Initialize robot: retrieve simulation timestep and robot characteristics
2. Set initial values (linear and angular velocities) and set robot's initial pose ($x_{robot}, y_{robot}, \phi_{robot}$)
3. Define global parameter values: N, β_2, Q
4. Compute angular sector: $\Delta\theta = \theta_{obs,2} - \theta_{obs,1}$
5. Initialize target
6. While target <= targetNr
    a) Exchange information with the simulator
    b) Get vehicle's pose, target position and simulation timestep
    c) Trigger a simulation step
    d) Processing step
        i. Set parameters values: $\tau_{tar}, \lambda_{tar}, Q$
        ii. For each sensor, compute the contribution of the repulsive forcelet
            A. Compute $\psi_{obs,i}$, $\lambda_{obs,i}$, σ_i
            B. Compute $f_{obs,i}$
            C. Compute $f_{obs} = f_{obs} + f_{obs,i}$

iii. Compute f_{obs} and f_{stoch}

iv. Compute resultant vector field f_{total} and assign it to angular velocity

v. Set linear velocity

vi. Define stop criterion for target distance (if desired)

e) View dynamics: plot the obstacle avoidance dynamics

f) Set robot's angular and linear velocities

7. Terminate simulation and cleanup

As a result, the following generic Matlab code was implemented (Listing 3.1):

```
%% IMPORTANT:
% Before running this script, open the scenario in VREP, e.g
% Do not run simulation!
%
%% Last change: Estela Bicho Erlhagen, 18/10/2018
%% run this with scenario MobileRobotDyn_TarObs.ttt

% Try to connect to simulator and initialize robot class:
vehicle = RobDyn_robot();

%get time step, dt, value (normally dt=50ms):
timestep = vehicle.get_simulation_timestep();

%Get robot physical characteristics:
[Rrobot,Theta_obs] = vehicle.get_RobotCharacteristics();

% Set initial velocity vrobot = 0 (linear velocity) and wrobot = 0 (angular velocity)
% and get robot's inicial pose
wrobot = 0.01; %rad/s
vrobot = 10; %cm/s
[xrobot, yrobot, phirobot] = vehicle.set_velocity(wrobot, vrobot);

% Define the (first) Target
ntarget=1; %initialize first target

%% 1. set global parameter values
% obstacles: beta1, beta2, delta_theta, Q
TOO_FAR = 75; % defines a radius for repulsive forces (cm)
N = 11; % nr of sensors
beta2 = 20; % flexible
```

```
delta_theta = Theta_obs(2) - Theta_obs(1);
Q = 0; % pre-factor of stochastic force, Q

%% 1.2 create vectors for psi_obs, lambda_obs, sigma
psi_obs = zeros(N,1);
lambda_obs = zeros(N,1);
sigma  = zeros(N,1);
fobs = zeros(N,1);

%%%---------------------- Start Robot Motion Behavior -------------------
%%%-----------------------------------------------------------------------
while ntarget<=vehicle.TARGET_Number % until robot goes to target ntarget

   %% Robot interface
   % set and get information to/from vrep
   % avoid do processing in between ensure_all_data and trigger_simulation
   vehicle.ensure_all_data();

   % Get vehicle's pose: (xrobot,yrobot) in cm, phirobot in rad
   [xrobot, yrobot, phirobot] = vehicle.get_vehicle_pose();

   %Sensorial information: get distance from obstacles (11 infrared obstacles - distance
       between
   %0-80cm):
   [d] = vehicle.get_DistanceSensorAquisition();  %distance in cm

   %get simulation time
   sim_time = vehicle.get_simulation_time();

   %trigger simulation step
   vehicle.trigger_simulation();

   %% Processing step
   %%----------- BEGIN YOUR CODE HERE ----------- %
   dt = timestep;
   tau_obs_min = 5 * dt; % flexible
   beta1 = 1/tau_obs_min; %  = lambda_obs_max = 1/tau_obs_min
   Fobs = 0;

   for i=1:N
   %% 2. for each sector compute the contribution of the repulsive forcelet
   % 2.1. compute repeller value
   psi_obs(i) = phirobot + Theta_obs(i);
```

```
73      % 2.2 compute magnitude of repulsion
        if d(i) < TOO_FAR
75          lambda_obs(i) = beta1 * exp( -d(i)/beta2 );
        else
77          lambda_obs(i) = 0;
        end

        %2.3. Compute range of repulsion
81      sigma(i) = atan( tan(delta_theta/2) + Rrobot / (Rrobot + d(i)) );
        % 2.4. Compute f_obs_i
83      fobs(i) = lambda_obs(i) * (-Theta_obs(i)) * exp( - Theta_obs(i)^2 / (2 * sigma(i)^2 ));
        Fobs = Fobs + fobs(i); % get all contributions
85      end

87      %% 4. compute stochastic force, stoch
        fstoch = sqrt(Q)* randn(1,1);

        %% 5. Compute resultante vector field
91      f_total = Fobs + fstoch;

93      %% 6. Compute angular velocity
        wrobot = f_total;

        %% 7. Set linear velocity
97      vrobot = 20; % cm/s

99      %% 8. Stop Criterion

101     %% view dynamics
        % visualization of the dynamics (target acquisition)
103     phi_plot = (0:10:360)*pi/180;
        lphi = length(phi_plot);
105     fstoch_plot = sqrt(Q) * randn(lphi, 1);
        f = figure(1);
107     f.Name = 'Obstacle Avoidance dynamics';
        f.ToolBar = 'none'; f.MenuBar = 'figure';
109     clf;
        hold on
111     fobs_plot_all = 0;
        for j = 1:N
113         % plot
            fobs_plot = lambda_obs(j) .* (phi_plot - psi_obs(j)) .* ...
115             exp(- (phi_plot - psi_obs(j)).^2/(2 * sigma(j) ^2));
```

```
        p = plot(phi_plot * 180/pi, fobs_plot);
117     p.LineWidth = 2;
        legStr{j} = ["$f_{obs, " + num2str(j) + "}$"];
119     % accumulate
        fobs_plot_all = fobs_plot_all + fobs_plot;
121 end
    legStr{N+1} = ["$f_{obs,all}$"];
123 p = plot(phi_plot * 180/pi, fobs_plot_all);
    p(1).LineWidth = 3;
125 p(1).Color = 'k';
    % axes customization
127 ax = gca;
    yl = ax.YLim; % get current axis y-axis limits
129 ax.XLim = [0, 360];
    ax.XTick = ax.XLim(1) : 45 : ax.XLim(2);
131 ax.Title.String = 'Obstacle avoidance dynamics: $F_{obs} (\phi)$';
    ax.XLabel.String = 'Heading direction - $\phi (^{\circ})$';
133 ax.YLabel.String = 'Magnitude';
    ax.Title.Interpreter = 'latex';
135 ax.YLabel.Interpreter = 'latex';
    ax.XLabel.Interpreter = 'latex';
137 ax.Title.FontSize = 14;
    ax.YLabel.FontSize = 14;
139 % Line
    phi_robot_plot = phirobot;
141 if(phirobot < 0)
        phi_robot_plot = phirobot + 2*pi;
143 end
    line([ phi_robot_plot * 180/pi, phi_robot_plot * 180/pi], [yl(1), yl(2)]);
145 % Legend
    lgd = legend(legStr);
147 lgd.Location = 'southeast';
    lgd.Interpreter = 'latex';
149 lgd.FontSize = 10;
    hold off
151 %%------------- END OF YOUR CODE -------------
    % set robot angular velocity (rad/s) and robot linear velocity (cm/s)
153 vehicle.set_velocity(wrobot, vrobot);
    %--------------------------------------------------------------------------
155 end
vehicle.terminate();
```

Listing 3.1: Generic implementation of obstacle avoidance behavior (not tuned)

3.2.1 Scenario 1

The obstacles were placed at a distance of 0 cm between each other to form a wall (no gap). The robot was placed in the middle of the arena and facing the wall at a distance of 100 cm. The stochastic force was reset (null variance – $Q = 0$). The robot moves at a constant linear velocity of 20 cm/s, which is a reasoable value. The parameter $\beta_2 = 20$ was fixed and β_1 was increased until the robot avoids collision with the wall. Then, for each value of β_1, the contribution of each sensor and the resultant obstacle avoidance dynamics were observed.

β_1 is related to $\min(\tau_{obs,i})$ (Eq. (3.12)), thus, **tau_obs_min** was set, starting from $10\Delta t$ and decremented at unit steps. Fig. 3.4 illustrates the simulation for **tau_obs_min** = $5\Delta t$, where a collision occurs with a wall (robot becomes red).

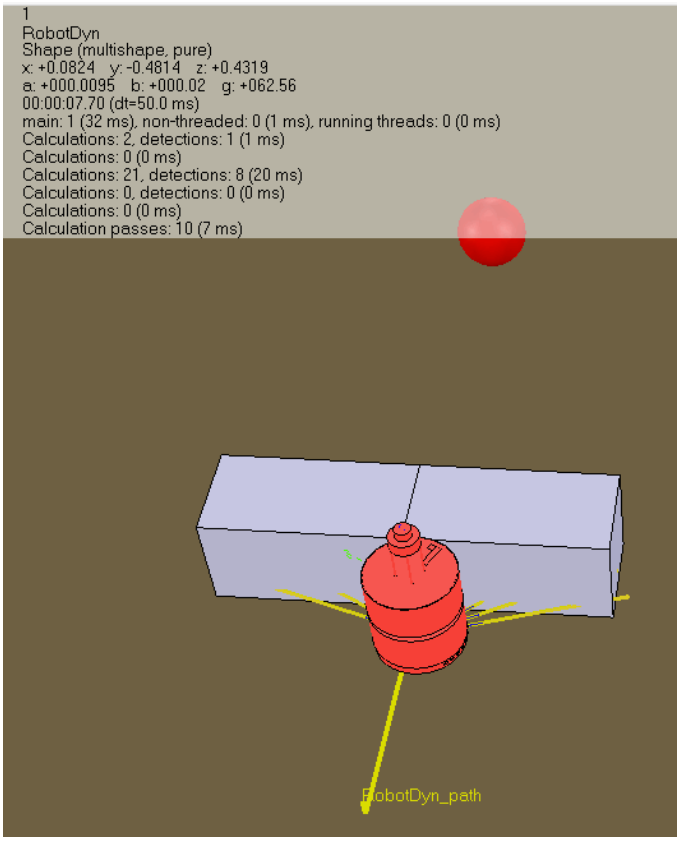


Figure 3.4: Obstacle avoidance behavior: **tau_obs_min** = $5\Delta t$

Conversely, in Fig. 3.5 is illustrated the final state of the first compliable value of $beta_1$ for collision-free path (**tau_obs_min** = $4\Delta t$), as indicated by the auxiliary text (see also Video ./videos/obs-2-1.mp4). In the obstacle avoidance dynamics plot it can be observed that, in the final state depicted, the heading direction is very far appart from the repeller, as expected.

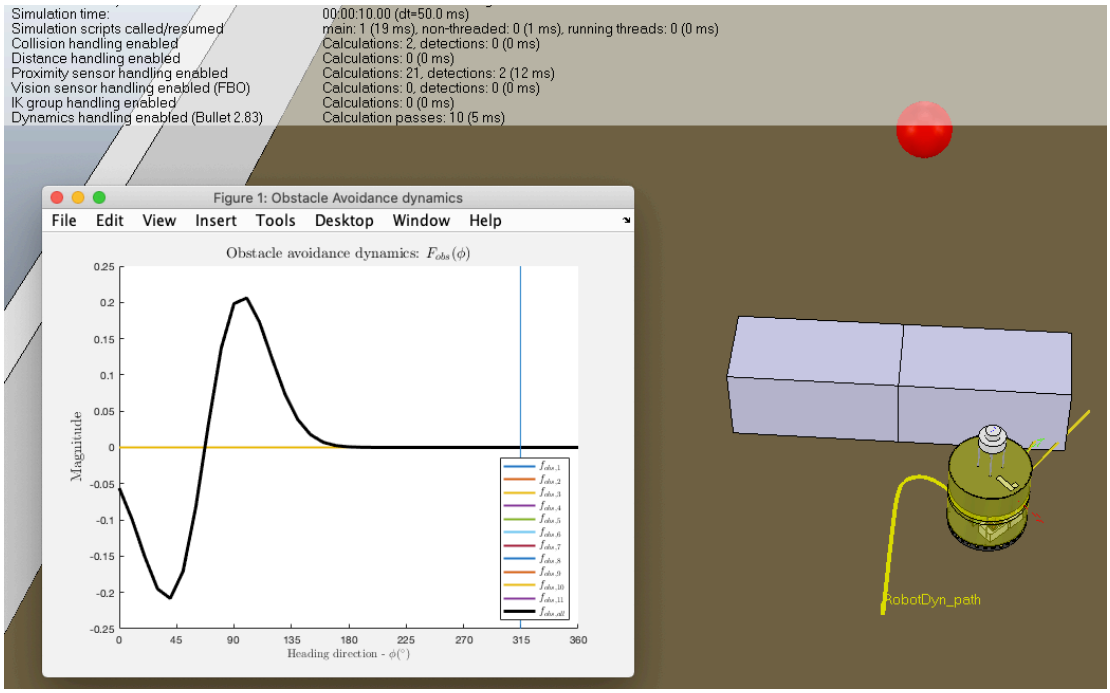


Figure 3.5: Obstacle avoidance behavior: **tau_obs_min** = $4\Delta t$ (final state)

The deviation from the obstacles only happened when the strenght of the repeller was sufficiently high to induce significant angular velocities, which for this case, started at $\omega = d\phi/dt = -0.5$ rad/s and hit its maximum at approximately $\omega = -3$ rad/s, representing an counterclockwise turn. It is important to note, however, that this calibration for collision-free path of the parameter β_1 is dependent on the value of the linear velocity, as the system may not have time to react to it — the higher the value of the linear velocity (constant), the lower the value of **tau_obs_min** needs to be. On the other hand, the linear velocity should decrease with the distance to the obstacles, which could be modelled as a dynamic system, e.g., the one in Eq. (2.10).

Fig. 3.6 shows in more detail the obstacle avoidance dynamics plot, used as an aiding tool, with the robot direction direction still facing the wall, i.e., $\phi = \psi_{obs}$. It can be seen that 5 sensors are detecting the obstacles, yielding a stronger reppeller in that direction, but still insufficient for significant direction change.

The addition of noise to the dynamic system was unnecessary, as there is a slight bias from sensors readings, due to uneven obstacle detection by all sensors — the leftmost sensors did not detect the obstacles initially, whereas the rightmost did. This slight bias for obstacles detection is responsible for the counter-clockwise turn, rendering it unnecessary to incorporate noise for the robot to deviate from the obstacles, albeit the heading direction initially sits in a repeller.

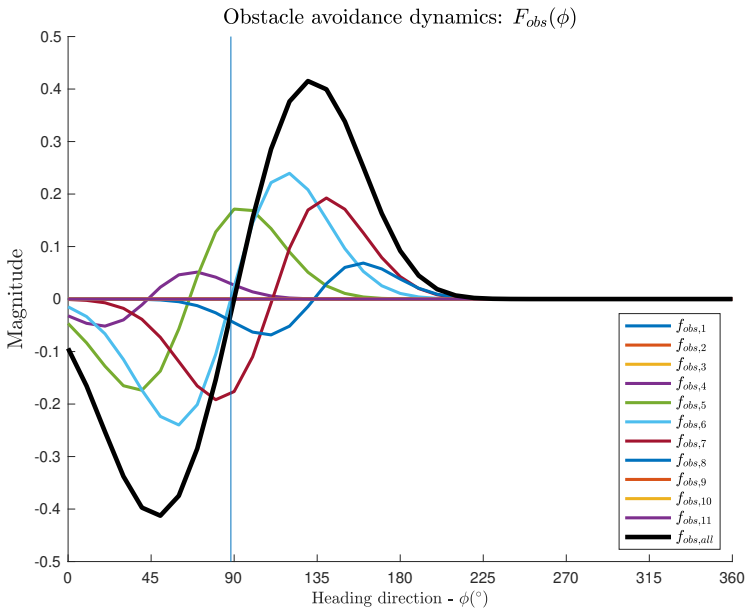


Figure 3.6: Obstacle avoidance behavior: dynamics — **tau_obs_min** = $4\Delta t$

3.2.2 Scenario 2

The obstacles were now placed at a 10 cm gap between each other, and the robot was placed in the middle of the arena facing the gap at a distance of 100 cm. The remaining conditions from scenario 1 remained unchanged.

First, the system was simulated with the previous value of β_1 to assess the system's behavior (Fig. 3.7). It can be seen that the robot collides with an obstacle in the attempt to deviate from it, as the heading direction still remains inside the influence of repellers.

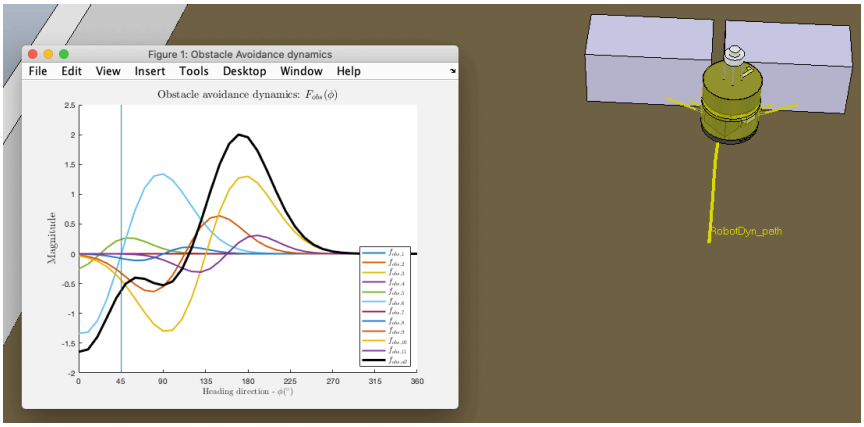


Figure 3.7: Obstacle avoidance behavior: dynamics — **tau_obs_min** = $4\Delta t$, gap = 10 cm

It does not detect obstacles in most of the path in the most central sensor, as it moves forward, due to the gap, which was the most strong contribution. When the contributions of the other sensors are high enough to trigger robot's rotation, the robot starts to rotate (counterclockwise, in this case), but the dynamics is not fast or strong enough to avoid the obstacle.

The value of parameter β_1 was then retuned to avoid collisions with the obstacles, following the same procedure, i.e., decrementing **tau_obs_min**. Fig. 3.8 illustrates the first collision-free path of the robot with the new environment conditions, occurring at **tau_obs_min** = $3.5\Delta t$ (see also Video ./videos/obs-2-2.mp4).

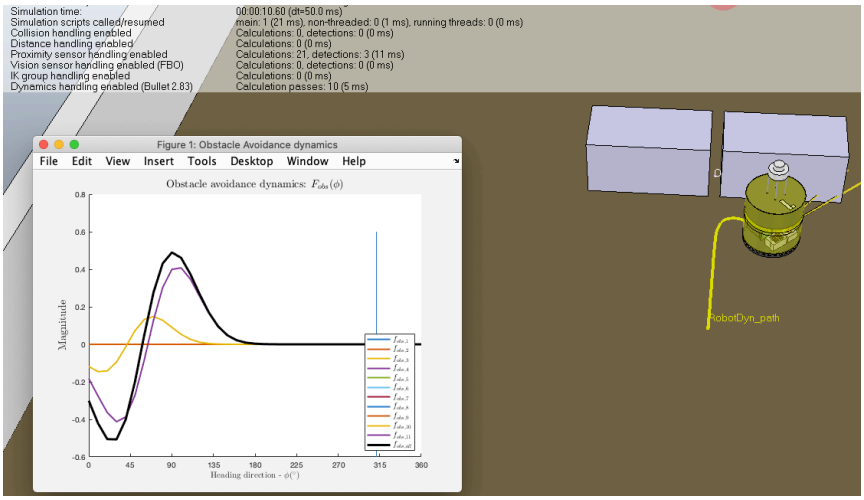


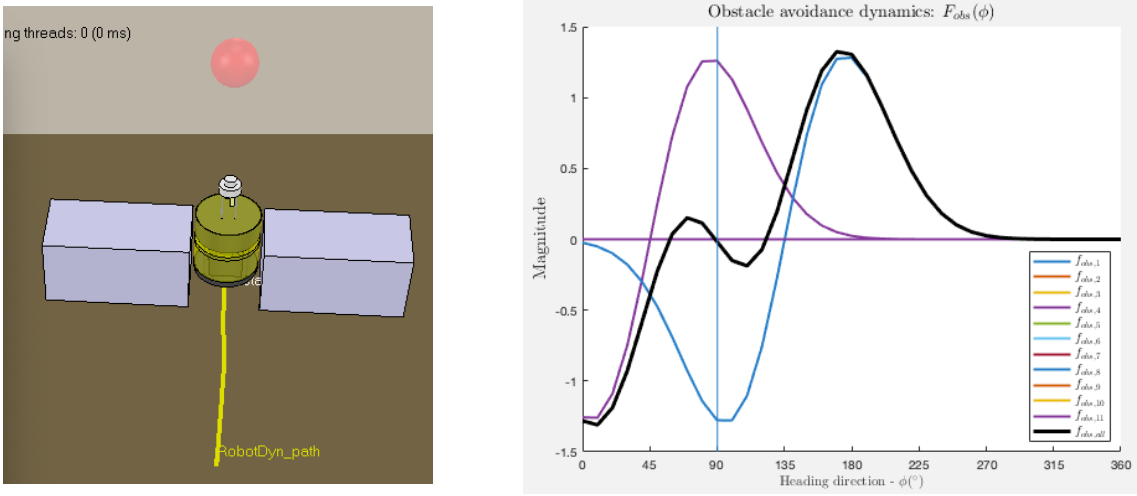
Figure 3.8: Obstacle avoidance behavior: dynamics — **tau_obs_min** = $3.5\Delta t$, gap = 10 cm

3.2.3 Scenario 3

In this scenario several simulations are performed for different gaps between obstacles, namely 20–80 cm, with a 10 cm span. Fig. 3.9 illustrates the first case where the robot can move between the obstacles, which occurs for a distance gap of 50 cm between them. (see also Video ./videos/obs-2-3-50cm.mp4). Fig. 3.9b shows that for a 50 cm gap, in the situation depicted in Fig. 3.9a, the heading direction has an attractor, instead of a repeller (0–40 cm), and two repellers at approximately 60° and 120°.

3.2.4 Bifurcation analysis

As indicated in Section 3.2.3, the heading direction dynamics exhibits different qualitative behavior — the stability of the fixed points varies — starting from gap = 50 cm. This corresponds to a bifurcation point,



(a) Simulation

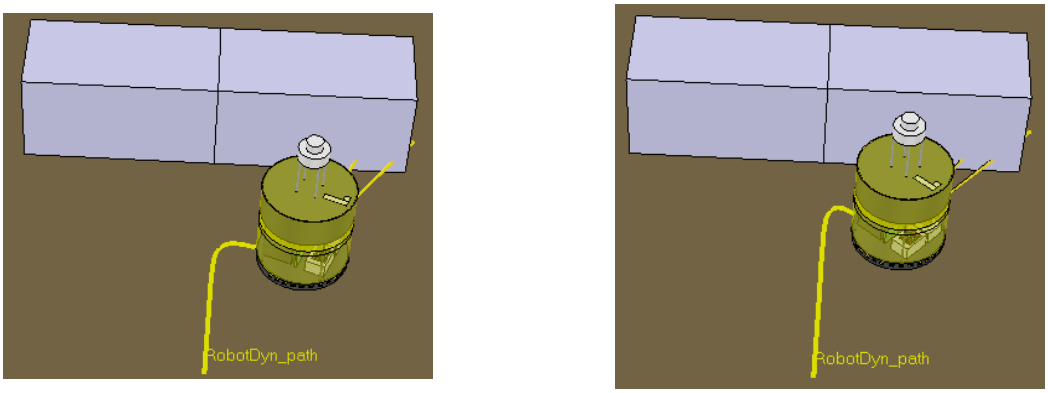
(b) Dynamics

Figure 3.9: Obstacle avoidance behavior: dynamics — **tau_obs_min** = $3.5\Delta t$, gap = 50 cm

representing the distance below which the robot (with diameter 45 cm) cannot pass between the two obstacles. For gap < 50 cm, the planning dynamics has an reppeller at the heading direction $\phi = \pi/2$, and for gap > 50 this reppeller becomes asymptotically stable (i.e., an attractor).

3.2.5 Influence of parameter β_2

The parameter β_2 is the decay rate of the repulsion force with the distance increase. Thus, to test this, β_2 was varied and simulated in the conditions of scenario 1, and the resulting repulsion magnitude $\lambda_{obs,i}$ computed (see Fig. 3.10).



(a) $\beta_2 = 50$

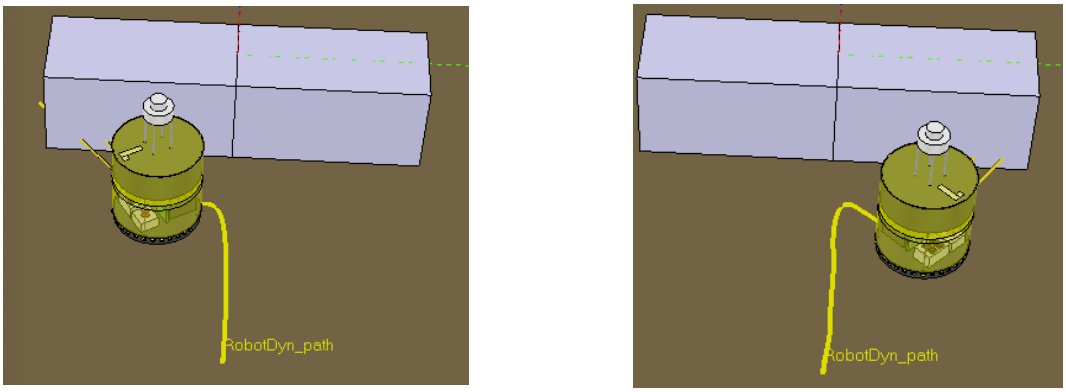
(b) $\beta_2 = 15$

Figure 3.10: Obstacle avoidance behavior: influence of parameter β_2

For increasing values of β_2 (Fig. 3.10a), the decay rate diminishes, maintaining the repulsive effect significantly for a wider distance range — the robot starts to rotate farther away from the obstacles. Conversely, for decreasing values of β_2 (Fig. 3.10b), the decay rate increases, maintaining the repulsive effect significantly for a narrow distance range — the robot starts to rotate closer to the obstacles.

3.2.6 Influence of noise

The influence of noise was tested, varying the value of the magnitude of gaussian white noise, Q (see Fig. 3.11).



(a) $\beta_2 = 15, Q = 0.05$ (b) $\beta_2 = 15, Q = 0.5$

Figure 3.11: Obstacle avoidance behavior: influence of noise

Comparing Fig. 3.11a with Fig. 3.10b, it can be seen that the mere introduction of noise induced a different qualitative behavior with the decision of turning left instead of right, respectively. Additionally, in Fig. 3.11b, it can be seen the magnitude of noise should be maintained fairly low, as it may introduce jitter into the system, depicted by the less 'clean' path. Thus, the noise should be sufficient to enough to guarantee the escape from the repellers within a time limit, in case the system is initially placed there.

3.2.7 Scenario 4

Fig. 3.12 illustrates scenario 4, composed of several obstacles that the robot must avoid. It can be observed that the robot is able to avoid all obstacles, as expected.

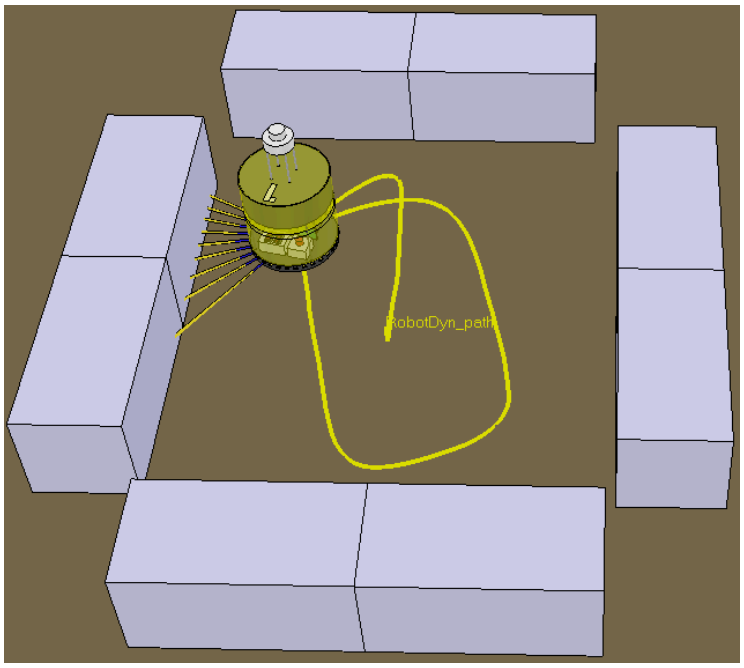


Figure 3.12: Obstacle avoidance behavior: Simulation with several obstacles — **tau_obs_min** = $3.5\Delta t$, Q = 0.05, $\beta_2 = 20$

4 Integration of behaviors: Obstacle Avoidance and Target Acquisition (nonlinear)

In this chapter the obstacle avoidance and target acquisition behaviors are integrated, but for the latter is considered a nonlinear dynamic system. A detailed analysis is then performed over: obstacle avoidance and target acqusition time constant precedence and tuning; distance between obstacles and respective bifurcation analysis; the influence of noise (stochastic force). Finally, more demanding scenarios are simulated, respectively in S or U shape, and the robot's behavior is analyzed.

4.1 Implementation

The nonlinear dynamic system that controls the movement of the robot for target acquisition in a collision-free path is obtained through the sum of the obstacles and target dynamic systems contribuitions and also a stochastic component for escaping repellers in a finite time, as given by Eq. (4.1):

$$\frac{d\phi}{dt} = \sum_{i=1}^{N} f_{obs,i}(\phi) + f_{tar}(\phi) + f_{stoch} \tag{4.1}$$

Thus, the implementation is simply the sum of these components (see Listing 4.1).

```
%% compute f_total
F_total = Fobs + ftar + fstoch;
```

Listing 4.1: Implementation of obstacle avoidance behavior and target acquisiton

However, as it will be discussed further ahead, the tuning of the overall system parameters is critical for adequate collision-free navigation to the target. By design, the obstacle avoidance behavior takes precedence over target acquisition, as obviously, in a limit case — e.g., where the target is behind an obstacle —

the robot must circumvent the obstacle, instead of hitting it. This is guaranteed by imposing higher magnitude for repulsive component than for attractive one, i.e. $\lambda_{obs} \gg \lambda_{tar}$, and consequently, $\tau_{obs} \ll \tau_{tar}$.

4.2 Scenario 1

In this section are analyzed the influences of varying the magnitude of attraction to target, λ_{tar}, the distance between obstacles, and the noise (stochastic force) in the robot's navigation behavior.

4.2.1 Magnitude of attraction to target: λ_{tar}

If the magnitude of the target is greater than the obstacles, the issue that occurs is that it is only when the robot is a very short distance from the obstacle that the repeller is established. When the robot is close to the obstacle this one is detected in several directions and the sum of these repulsive forces exceeds the force of attraction, changing the stability of the fixed point of that navigation direction (Fig 4.1).

Consequently, as shown in Fig. 4.1, or the robot collides with the obstacle or has a very irregular route, depending on the time constant. Having enough time to escape the repeller, the robot has a zigzag route, because it approaches the obstacle and moves away successively since it is only close to this that it realizes that there is an obstacle. If there is not enough time to change the heading direction, the robot collides. Therefore, the magnitude of the repulsing from the obstacle must always be greater than that of the attraction to the target, i.e., $\lambda_{obs} \gg \lambda_{tar}$, and consequently, $\tau_{obs} \ll \tau_{tar}$.

4.2.2 Different gaps between obstacles

In this scenario, **MobileRobotDyn_Tar_Obs.ttt**, several simulations are performed for different gaps between obstacles, namely 0 cm, 10 cm and 50 cm span.

4.2.2.1 No gap (0 cm)

Fig. 4.2 illustrates the simulations performed for obstacles forming a wall without gap (0 cm).

In the first simulation, the obstacles form a wall and the robot is positioned in front of the obstacles but without the sensors detecting it. (Fig. 4.2a). The stochastic force is not considered and the parameter β_2 is 50. In the initial moments, obstacles are not detected by the sensors, so they do not contribute to the dynamics vector field in the navigation direction positioned in front of the obstacles but without the sensors detected.

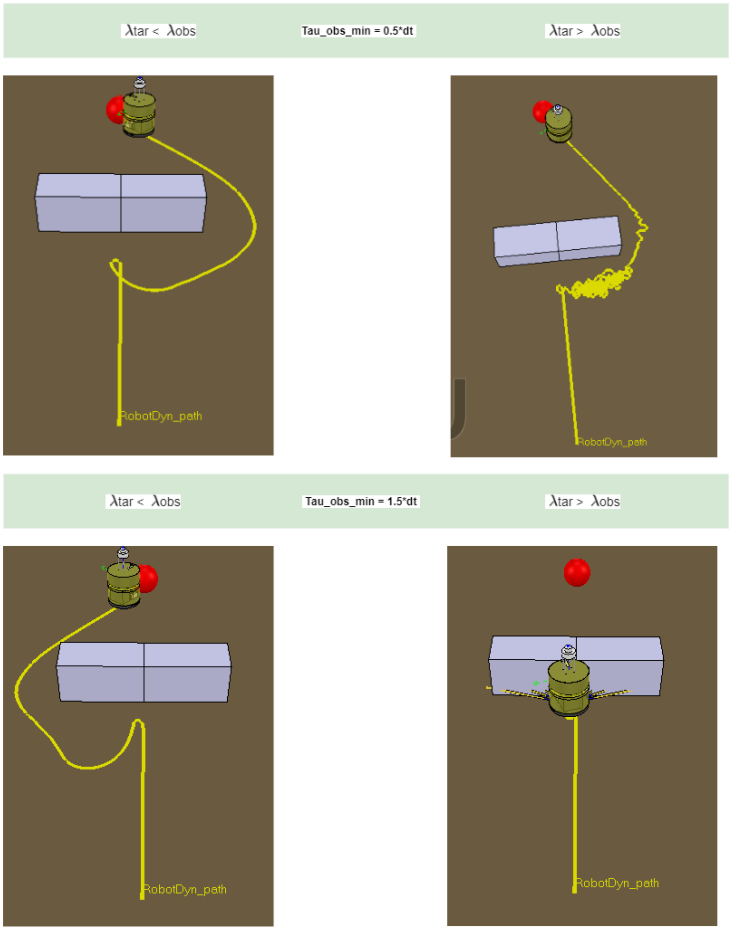


Figure 4.1: Different λ_{tar} values simulations results.

As the robot approaches the wall, the sensors begin to detect obstacles, in the first moments, due to the considerable distance to the obstacle, the repeller is of low magnitude so it does not prevail, therefore an attractor is formed in the direction of navigation, with a lower intensity of attraction than before. However, as the distance to the obstacles decreases, it is visible that the intensity of the repeller increases and dominates, an instability occurs, the vector field of the dynamics of the navigation direction starts to have a repeller in that direction. When the stability of the fixed points changes, it because the bifurcation point has been exceeded, and the robot will present a different behaviour than previous.

So such behavior is in line with expectations since with the proximity to an obstacle the forces of repulsion appear and prevail over attraction forces. Because of that the robot can escape from the obstacle to the right or left, because there are two attractors in these directions, attending at the Figs. 4.2b to 4.2e, it can be noted, the heading direction of the robot converges to one of the attractors. In this simulation episode, the robot moves to the right to avoid a collision with the wall.

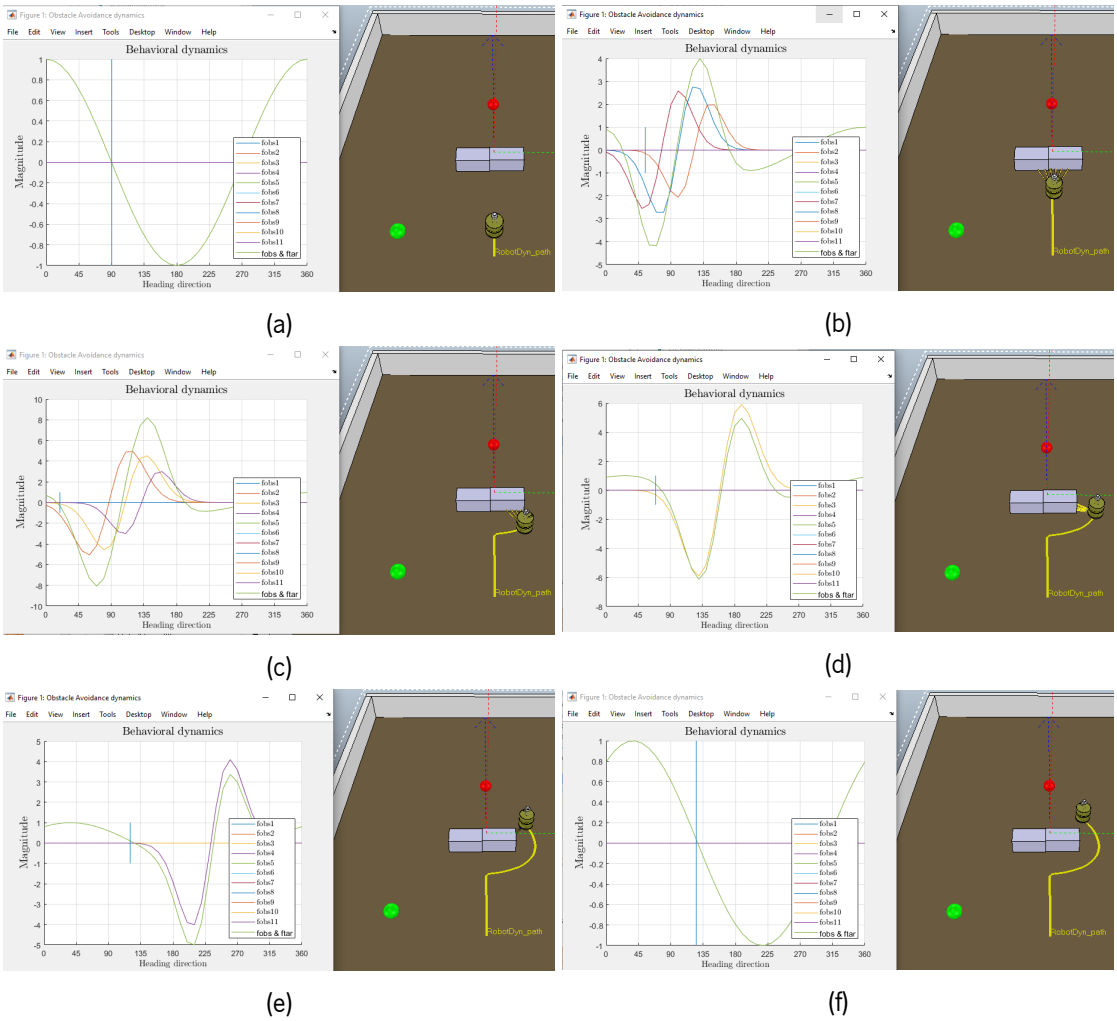


Figure 4.2: Behavioral Dynamics depending on the robot's position in the world

When the obstacle is surpassed, it is no longer detected by the sensors, the repulsion forces sum is zero, so they no longer contribute to the vector field. Because of that, there is an attractor in the direction of the target, and this is where the robot progresses successfully. Once again, the robot's behavioral dynamics changed. (Fig. 4.2f).

4.2.2.2 10 cm

In the second simulation, the obstacles are 10 cm apart and the robot is positioned in front of the obstacles but without the sensors detecting them, as illustrated in Fig. 4.3. The stochastic force is not considered, the parameter β_2 is 50.

In Fig. 4.3a the sensors have not yet detected obstacles, so only the target contributes to the vector field,

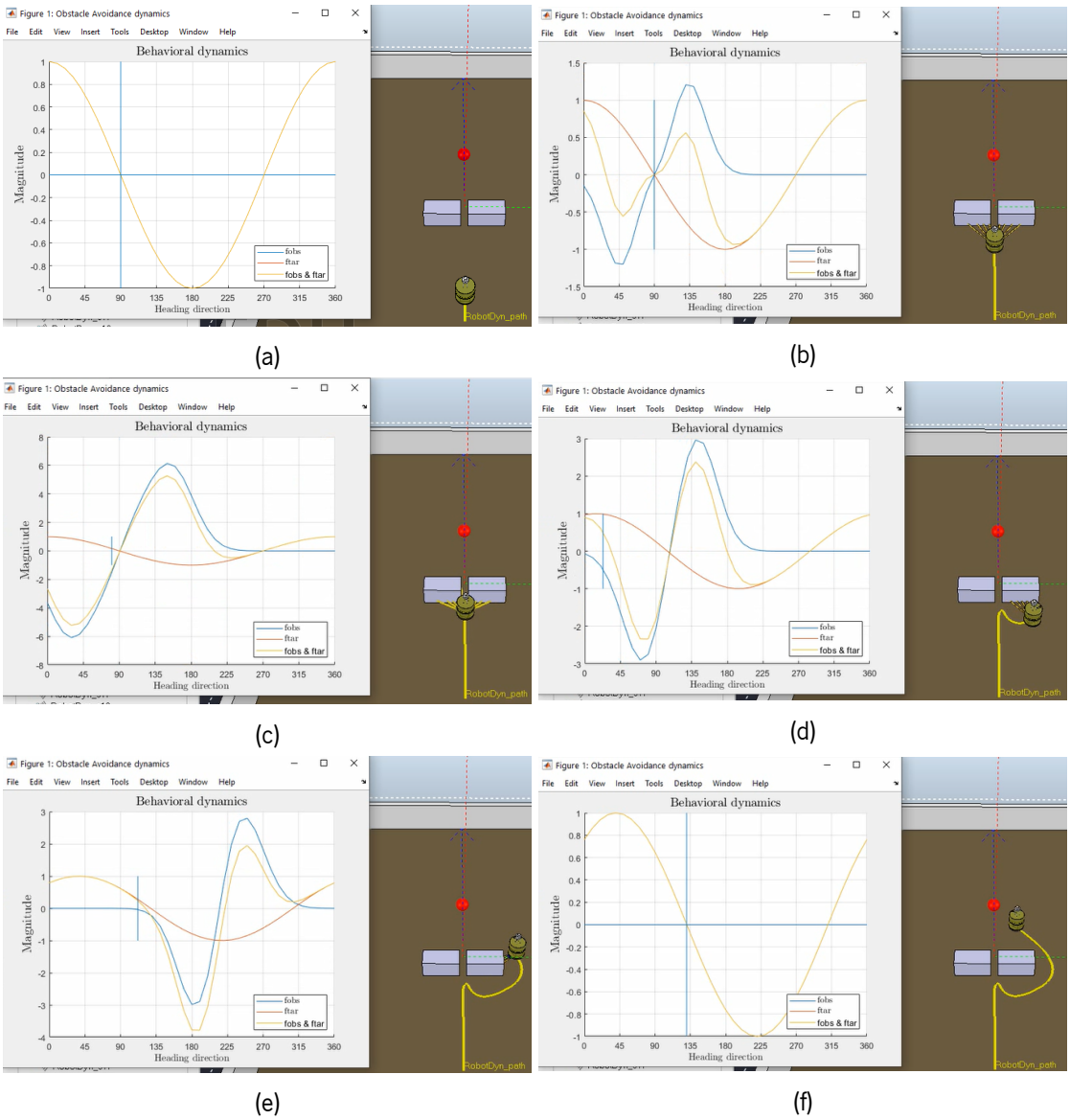


Figure 4.3: Behavioral Dynamics depending on the robot's position in the world for a 10 cm gap

yielding an attractor in the forward heading direction. The robot proceeds. As the robot approaches the obstacles, the repulsion forces present greater magnitude, and therefore dominate, yielding a repeller in the heading direction of 90 degrees, and more, two attractors are erected, one on the left and one on the right, giving two escape routes for the robot so that there is no collision with the obstacles. It is in this situation that the behavior change occurs — the fixed point for the heading direction was asymptotically stable and now becomes unstable, causing the robot to turn, in this case, to the right (see Figs. 4.3b and 4.3c).

Figs. 4.3d and 4.3e show, through the diagram, that the robot progresses in the direction of one of the attractors, and through the simulation environment, it is seen that the robot bypasses the obstacle without

collisions, because the vector field continues with a repeller in the direction of the obstacles. With the movement towards the attractor, the heading direction of the robot meets the target, as desired.

4.2.2.3 50 cm

In the last simulation, the obstacles are 50 cm apart and the robot is positioned in front of the obstacles but without the sensors detecting them as illustrated in Fig. 4.4. The stochastic force is not considered and the $\beta_2 = 50$.

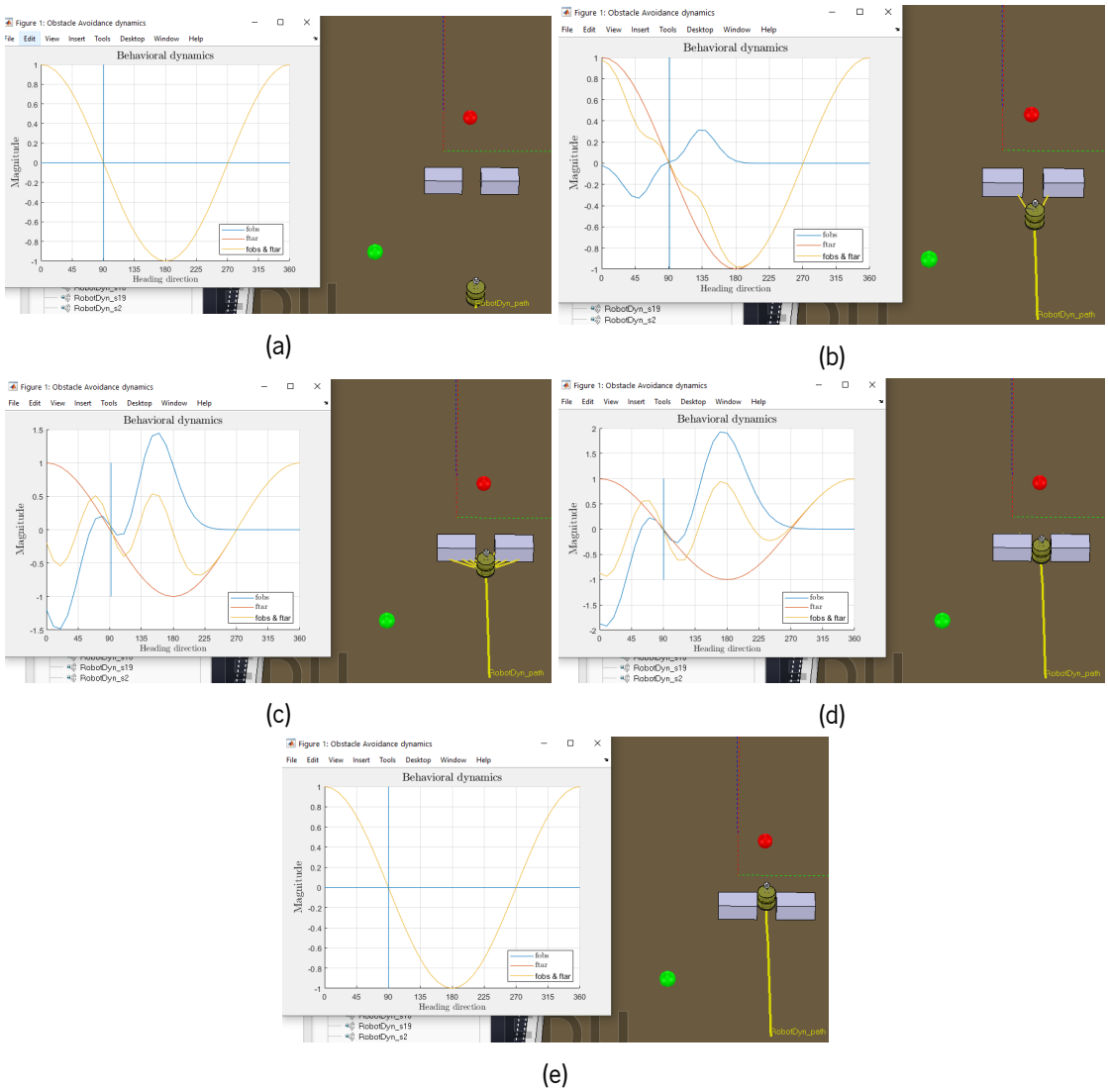


Figure 4.4: Behavioral Dynamics depending on the robot's position in the world for a 50 cm gap

In the beginning (Fig. 4.4a), no obstacle is detected – the vector field of the dynamics of the heading

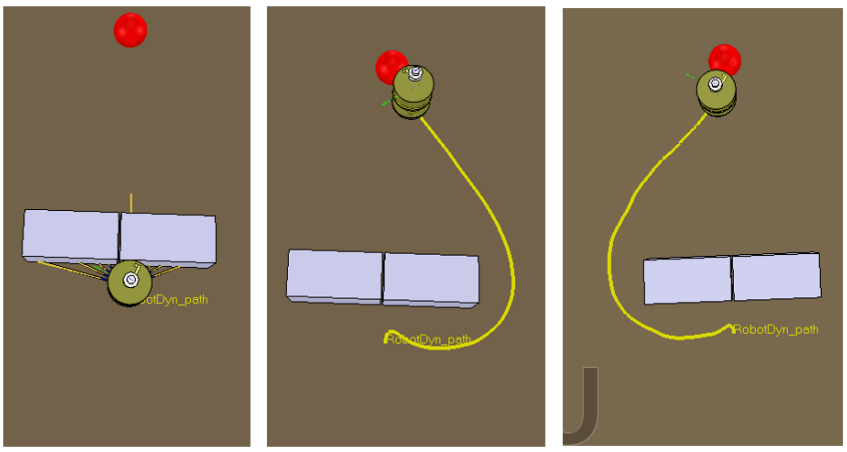
direction is an attractor in the direction of the target. As the robot continues forward, the target's contribution remains, with an attractor in the 90-degree heading direction. In the first moments when the robot's sensors detects obstacles (Fig. 4.4b), the sum of its forces yields a repeller at 90 degrees, where an attractor already exists, a contribution made by the target. It is clear by the figure that the contribution made by the target has greater magnitude, so it dominates, therefore in the heading direction of 90 degrees an attractor is established. The robot proceeds forward.

With the proximity to the obstacles (Fig. 4.4c), it is increasingly noticeable that the obstacles are spaced apart, so each contribution of the obstacles will be more precise and, consequently, more distributed by the different degrees of heading direction, because of this the sum of all repulsion forces yields an attractor in the 90-degree navigation direction. What coincides with the attractor built by the force of attraction, the robot continues forward.

The vector field persists with this logic (Fig. 4.4d). When the robot overcomes obstacles, the sum of these repulsive forces becomes null and insignificant for the vector field (Fig. 4.4e).

4.2.3 Influence of noise

To verify the contribution of noise, the robot was positioned at a short distance from the wall, in two episodes noise was considered (Q = 0.05 and Q = 0.5) and in another it was not taken into account (Q = 0). It was found that in the situation in which the noise is added to the dynamics the robot manages to escape and avoid the obstacle, in the opposite situation the robot collided, illustrated in Fig. 4.5. This experiment reinforces the importance of adding noise to the dynamic system, for when the robot is in a repeller be able to escape in a finite time.



(a) Q = 0 (b) Q = 0.05 (c) Q = 0.5

Figure 4.5: Simulation results for the overall dynamic field with different noise levels

In addition, comparing the two experiments in which the noise was added it is also possible to make conclusions. The trajectory that the robot does when Q = 0.05 is smoother than when Q = 0.5. That said, when choosing the values of Q, it must be taken into account that the value must ensure that the robot escapes to the repeller and does not make abrupt direction adjustments, so always give preference to small values. Since the Gaussian white noise is a stochastic function, then the trajectory taken by the robot — the left or right attraction basin — will be selected in a probabilistic way by this function, contributes for behavioral diversity when the robot must deviate from the obstacles, enabling it to turn left or right, depending on the stochastic effect.

4.2.4 Scenario 2

In this simulation is used the **MobileRobotDyn_Tar_Obs_S.ttt** scenario, an S-shape scenario, to demonstrate the overall dynamics robustness for a more complex scenario. As can be observed (Fig. 4.6), the robot is able to avoid obstacles and reach the target.

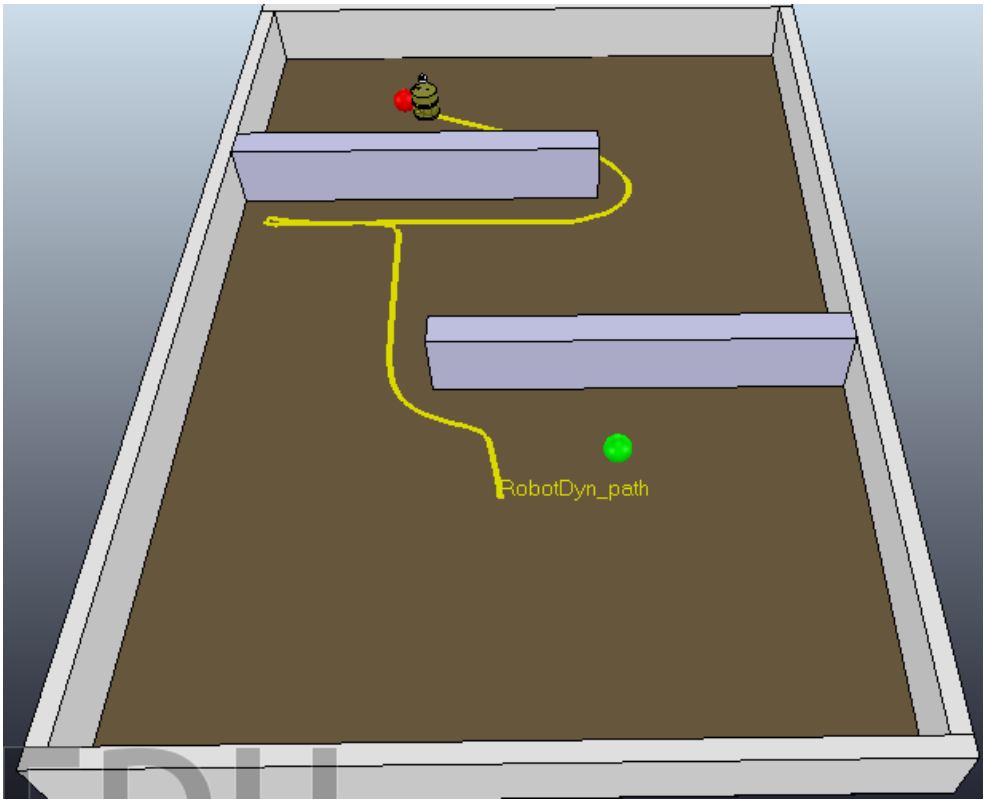


Figure 4.6: Simulation in the **MobileRobotDyn_Tar_Obs_S.ttt** scenario

4.2.5 Scenario 3

In this simulation is used the **MobileRobotDyn_Tar_Obs_U.ttt** scenario, an U-shape scenario, to demonstrate the overall dynamics robustness for an ever more complex scenario. (see Fig. 4.7 and Video ./videos/obs-tar-nonlinear-Uscenario.mp4). Once again, the robot is able to avoid obstacles and reach the target. The behavior is the same as already described: in the presence of an obstacle, a repeller is placed in that heading direction and two attractors, representing the possible heading directions for the escape, only when these forces are the most intense, which depends on the proximity to the obstacle. When the sensors do not detect obstacles, only the force of attraction to the target contributes to the vector field of the heading direction.

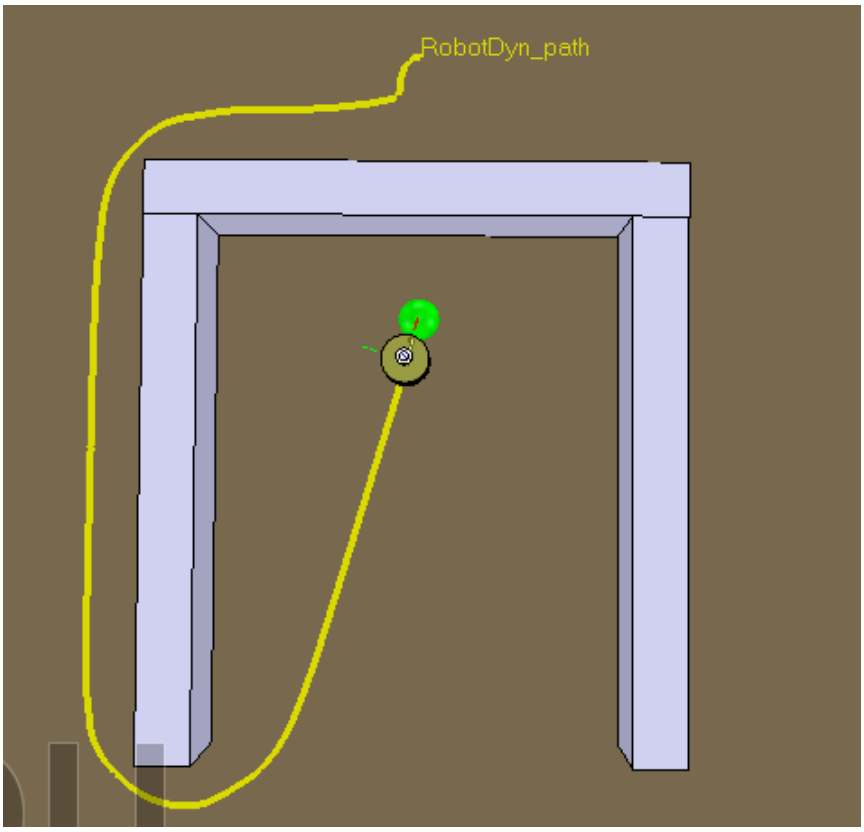


Figure 4.7: Simulation in the **MobileRobotDyn_Tar_Obs_U.ttt** scenario

4.2.6 Discussion

In conclusion, the vector field of the navigation direction is constituted by the attraction force and the sum of repulsion forces, with each force establishing an attractor and/or a repeller. With the movement of the

robot around the world, it is expected that it will change its behavior in accordance with its surroundings. The decisions that the robot undertakes to navigate with a certain heading direction or to continue in it depends on the bifurcations. Therefore, the stability of fixed points can change and other new fixed points appear.

The obstacle avoidance behavior must take precedence over the target acquisition to prevent the robot from hitting an obstacle while attempting to move to the target. This is guaranteed by imposing greater magnitude for repulsive component than for attractive one, i.e. $\lambda_{obs} \gg \lambda_{tar}$, and consequently, $\tau_{obs} \ll \tau_{tar}$.

In experiments with distances between obstacles, it should be noted that the bifurcation point slightly exceeds the diameter of the robot. This means that the vector field has an attractor in the direction of navigation 90 degrees only when the distance between obstacles is greater than the diameter of the robot, otherwise it will be a repeller in this direction and two attractors for the escape.

Finally, the stochastic force, here modelled as Gaussian white noise, is important to ensure escape from repellers within a finite time, which would otherwise remain in this state, provided this was the initial condition. Additionally, noise also contributes for behavioral diversity when the robot must deviate from the obstacles, enabling it to turn left or right, depending on the stochastic effect.

5 Integration of behaviors: Obstacle Avoidance and Target Acquisition (linear)

In this chapter the obstacle avoidance and target acquisition behaviors are integrated, but for the latter is considered a linear dynamic system. The main differences in the dynamics and corresponding behavior are discussed. Finally, the best dynamic system form for target acquisition behavior — nonlinear or linear — is discussed.

5.1 Implementation

The implementation of the dynamic system that controls the heading direction is yielded by replacing the nonlinear dynamic system for target acquisition — $f_{tar}(\phi)$ — for the linear one, as given by Eq. (2.17), in Eq. (4.1).

5.2 Simulation

In this scenario, **MobileRobotDyn_Tar_Obs.ttt**, are performed two simulations for different gaps between obstacles, namely 0 cm and 50 cm span.

5.2.1 No gap (0 cm)

In the first simulation (Fig. 5.1), the obstacles form a wall and the robot is positioned in front of the obstacles but without the sensors detecting them. The stochastic force is not considered and the parameter β_2 is 50.

In the initial moments, the sensors do not detect the wall, so the vector field of the navigation direction is constituted by an attractor, and the dynamic is linear, since the repulsive forces are zero and the attraction force is linear. The 90 degrees fixed point is the attractor and there are no repellers, the robot moves forward (Fig. 5.1a).

When obstacles are detected, the repulsion forces appear and their intensity exceeds the attraction force intensity, so in the direction of navigation 90 degrees, a repeller is established. Given the mathematical nature of the repulsive forces, two attractors are placed, one on the right and the other on the left, identifying the two directions of escape from the obstacle, (Fig. 5.1b).

In Figs. 5.1c and 5.1d the robot turns around itself, and proceeds the navigation direction to one of the attractors, in this case, it turns right. The robot continues to circumvent the obstacle until the sensors stop detecting it, so the behavior falls back to the initial one and the robot reaches the target (Figs. 5.1e and 5.1f).

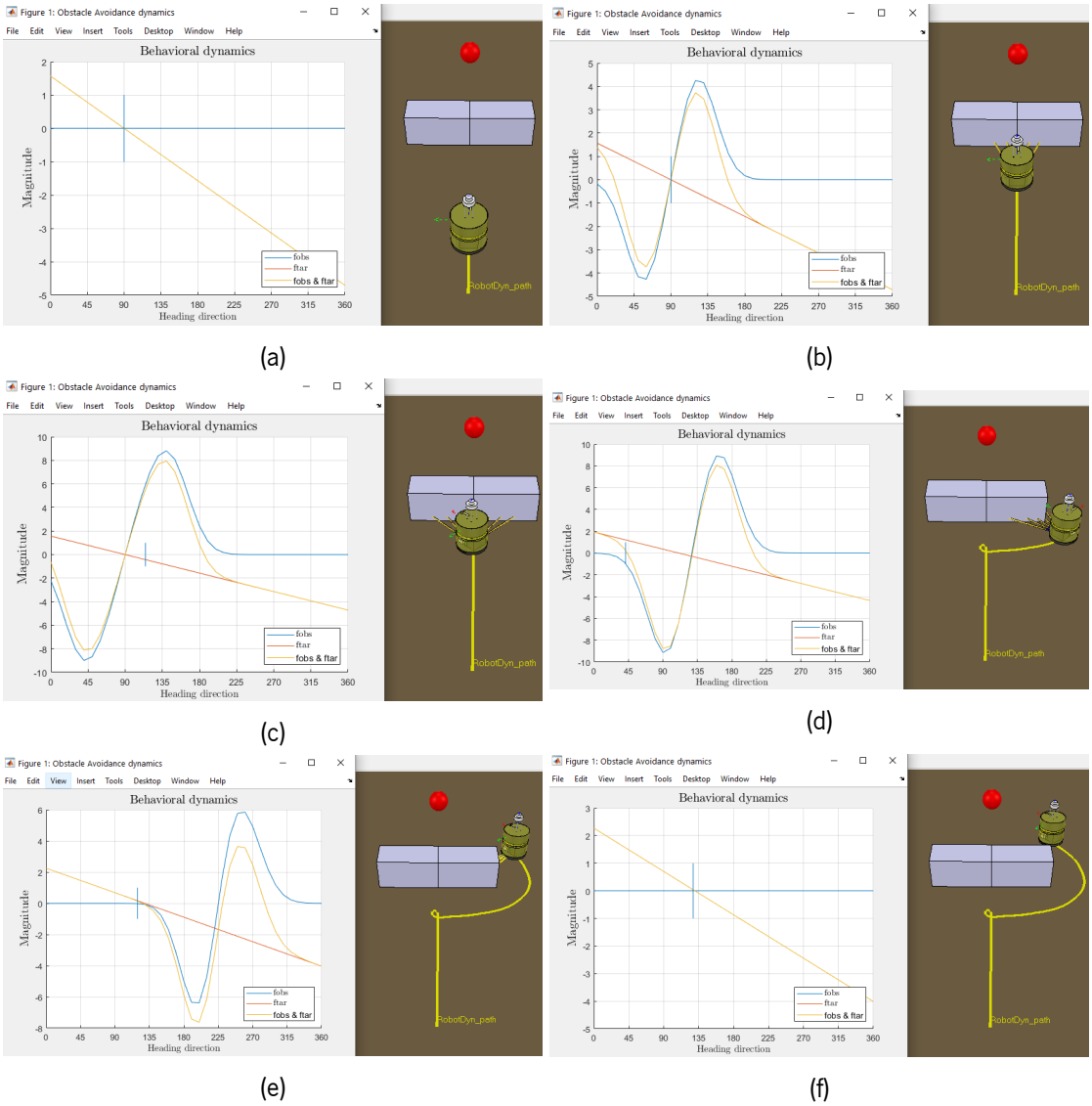
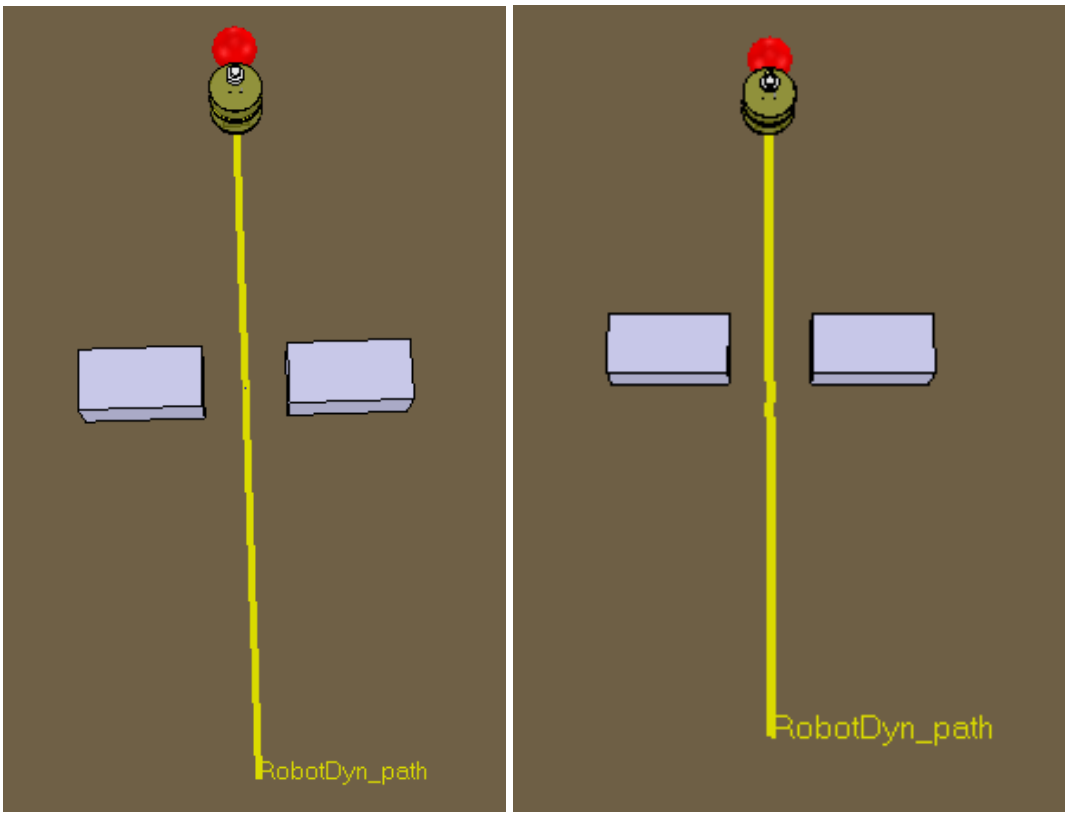


Figure 5.1: Behavior of the linear system at no gap between obstacles.

5.2.2 50 cm

A second experiment was carried out in the same scenario and with the same parameters in which the robustness of both systems, linear and non-linear, were evaluated (see Fig. 5.2 and Video ./videos/obstar-linear.mp4). Both reached the target through the shortest path between obstacles. However, the linear system showed more abrupt direction adjustments.



(a) Linear (b) Nonlinear

Figure 5.2: Comparison of linear and nonlinear dynamic systems for target acquisition in the overall dynamics at a 50cm gap between obstacles

5.3 Discussion

After analyzing the robot's behavior for both nonlinear and linear dynamics, it is now possible to compare them. Mathematically, the fact that the nonlinear function is a sinusoidal function, yielding always two fixed points — one attractor, and in the opposite direction a repeller — whereas the linear function only contributes with one attractor to the vector field. The repeller in the opposite direction to the target reinforces the heading

direction that must be taken by the robot so that it reaches the target. In the previous experiments, as well by Section 2.4.5, it is concluded by experimentation too, that the nonlinear system provides smoother paths at narrower angular velocity range. Thus, it can be concluded that the nonlinear dynamic system is better for the target acquisition behavior of the robot.

6 Control of driving speed (Extra)

As aforementioned, the planning dynamics is affected by the rate of fixed points local shifts, as the system must be able to track the attractor as it shifts, which effectively means that it is dependent on the path velocity, v. Effectively, planning dynamics attractors and repellers are established as the robot moves through the environment and sensory information changes or due to environmental changes (obstacles moving in the world). To keep the system stable, i.e. in or near an attractor at all times, the rate of such shifts must be limited to permit the track the attractor as it shifts. One way of accomplishing this is by controlling the path velocity of the vehicle, as the rate of fixed points shift is determined by the relative velocity of the robot with respect to its environment [1].

In this chapter, a more adequate dynamic system for path velocity is established — enhancing overall dynamics performance — and analysed, assessing its performance in several scenarios.

The maximal rate of shift of the fixed points as a function of the vehicle's velocity is given by [1]:

$$\dot{\psi}_{max} \approx \frac{\Delta\psi}{\Delta t} \approx \frac{v}{d} \tag{6.1}$$

This approximate description can be turn around to compute the desired path velocity as a function of distance with $\dot{\psi}_{max}$ as design parameter, that can be tuned to obtain good tracking. The desired velocity is computed separately for each of the two constraints ($j = \{tar, obs\}$) [1]:

$$V_j = d_j \dot{\psi}_{max} \tag{6.2}$$

The desired velocities are imposed through a very simple dynamics [2]:

$$\frac{dv}{dt} = -c_{obs}(v - V_{obs}) \exp\Big(-\frac{(v - V_{obs})^2}{2\sigma_v^2}\Big) - c_{tar}(v - V_{tar}) \exp\Big(-\frac{(v - V_{tar})^2}{2\sigma_v^2}\Big) \tag{6.3}$$

The strengths, c_{obs} and c_{tar}, are tuned such that in the presence of strong obstacle contributions the obstacle term dominates while in the absence of such contributions the reverse holds. A systematic way to construct a function that indicates if obstacles contributions are present, is to integrate force-lets, from

which a potential function of the obstacle avoidance dynamics results [1]:

$$U(\phi) = \sum_{i=1}^{N} \left(\lambda_i \sigma_i^2 \exp\left(-\frac{(\phi - \psi_i)^2}{2\sigma_i^2} \right) - \lambda_i \frac{\sigma_i^2}{\sqrt{e}} \right) \tag{6.4}$$

Positive values of this potential function indicate that the heading direction is in a repulsion zone of sufficient strenght, λ_i, so $c_{obs} > 0$ and $c_{tar} = 0$ is required. Conversely, negative values of the potential indicate that the heading direction is outside the repulsion range or repulsion is weak, so now $c_{obs} = 0$ and $c_{tar} > 0$ is required [1]. Effectively:

$$U(\phi) = \begin{cases} < 0, & c_{obs} = 0 \wedge c_{tar} > 0 \quad \rightarrow \quad \text{attractor} \\ > 0, & c_{obs} > 0 \wedge c_{tar} = 0 \quad \rightarrow \quad \text{repeller} \end{cases} \tag{6.5}$$

The transformation of potential levels to the strengths of the two contributions to the velocity control makes use of a sigmoidal threshold function [1]:

$$\alpha(\phi) = \frac{\arctan(cU(\phi))}{\pi} \tag{6.6}$$

ranging from $-1/2$ to $1/2$ (see Fig. 6.1, withdrawn from [1]).

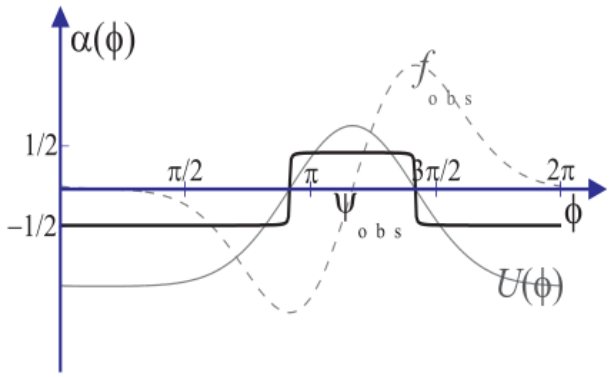


Figure 6.1: The dashed line is a repulsive force-let, f_{obs}. Its integral provided a potential (solid thin line), U, which is maximal near the heading direction to be avoided, i.e., resultant repeller. The thresholded potential (solid bold line), α, serves as an indicator of those intervals of the heading direction from which obstacle forces repel (withdrawn from [1])

Finally, the following functions for the strenghts of the two velocity contributions can be written:

$$\begin{aligned} c_{obs} &= c_{v,obs}(1/2 + \alpha(\phi)) \\ c_{tar} &= c_{v,tar}(1/2 - \alpha(\phi)) \end{aligned} \tag{6.7}$$

At sufficiently sharp sigmoids (C sufficiently large) this leads to the required transition behavior. The parameters, $c_{v,tar}$ and $c_{v,obs}$, determine the relaxation rate of the velocity dynamics in the two cases when either the obstacle or the target constraints dominate [1].

The following hierarchy of relaxation rates ensures that the system relaxes to the attractors and that obstacle avoidance has precedence over the target contribution [1]:

$$\lambda_{tar} \ll c_{v,tar}, \quad \lambda_{obs} \ll c_{v,obs}, \quad \lambda_{tar} \ll \lambda_{obs} \tag{6.8}$$

6.1 Implementation

Following the guidelines presented in the previous section, the overall dynamics with control of the driving speed was implemented. The first order differential equation for path velocity — Eq. (6.3) — was converted into an algebraic recursive equation, using forward Euler's method.

The potential function, $U(\phi)$, and thresholded potential function, $\alpha(\phi)$, were computed as an intermediate step. It should be noted that if no contributions from obstacles are present, then $\alpha(\phi) = 0$, yielding both obstacles and target contributions to the path velocity, i.e., that are two attractors for the dynamics. However, in the beginning the initial velocity of the robot is very low, resulting in a neglectable aceleration, thus causing the robot to move at this constant velocity until it detects the presence of obstacles. To prevent this, if $\alpha(\phi) = 0$, the robot moves at a reasonable constant speed.

Additionally, to assist the tuning of the parameters and the comprehension of the overall dynamics with control of the driving speed, the potential, $U(\phi)$, and thresholded potential ($\alpha(\phi)$) functions are plotted alongside with the obstacle avoidance and target acquisition dynamics.

As a result, the following generic Matlab code was implemented (Listing 6.1):

```
%%----------- BEGIN YOUR CODE HERE ----------- %
%% 1. set parameter values
dt = timestep;
% hierarchy of relaxation rates
% lambda_tar << c_v_tar
% lambda_obs << c_v_obs
% lambda_tar << lambda_obs
```

```
TOO_FAR = 75; % defines a radius for repulsive forces (cm)
N = 11; % nr of sensors
beta2 = 20; % flexible
delta_theta = Theta_obs(2) - Theta_obs(1);
Q = 0.05; % pre-factor of stochastic force, Q
% obstacle
tau_obs_min = 3.5 * dt; % flexible
beta1 = 1/tau_obs_min; %  = lambda_obs_max = 1/tau_obs_min
% target
tau_tar = 20*tau_obs_min; % relaxation time for attractor
lambda_tar = 1/tau_tar; % attraction magnitude
% accumulators
Fobs = 0; % accumulator for obstacle force
Upot = 0; % accumulator for Lyapunov´s function
% Lyapunov
Rtar = 20; % target radius
d_min = Rrobot + Rtar + 5; % minimum distance to target
C = 100; % potential constant
d_obs = d_min; % minimum distance to obstacle
d_tar = d_min; % minimum distance to target
psi_max = pi/12; % maximum rate of fixed points shift
sigma_v = pi/6; % angular range for velocity
% velocity
c_v_obs = 10 * beta1; % obstacle contribution to velocity
c_v_tar = 100 * lambda_tar; % target contribution to velocity
Vobs = d_obs * psi_max; % maximum velocity for obstacle avoidance behavior
Vtar = d_tar * psi_max; % maximum velocity for target acquisition behavior

%% 2. for each sector compute the contribution of the repulsive forcelet
for i=1:N
    lambda_obs(i) = 0;
    % 2.1. compute repeller value
    psi_obs(i) = phirobot + Theta_obs(i);
    % 2.2 compute magnitude of repulsion
    if d(i) < TOO_FAR
        lambda_obs(i) = beta1 * exp( -d(i)/beta2 );
    end
    %2.3. Compute range of repulsion
    sigma(i) = atan( tan(delta_theta/2) + Rrobot / (Rrobot + d(i)) );
    % 2.4. Compute f_obs_i
    aux = lambda_obs(i) * exp( - Theta_obs(i)^2 / (2 * sigma(i)^2 ));
    fobs(i) = aux * (-Theta_obs(i));
```

```
        % 2.5. Compute Lyapunov Function
52          k_i = lambda_obs(i) * sigma(i)^2 / sqrt(exp(1));
            upot(i) = aux * sigma(i)^2 - k_i;
54          % 2.6. Compute all obstacles contributions
            Upot = Upot + upot(i);
56          Fobs = Fobs + fobs(i);
    end
58  %% 3. compute psi_tar
    psi_tar = atan2(YTARGET - yrobot, XTARGET - xrobot);
60  %% 4. compute ftar
    ftar = -lambda_tar * sin(phirobot - psi_tar);
62  %% 5. compute stochastic force, stoch
    fstoch = sqrt(Q)* randn(1,1);
64  %% 6. Compute resultante vector field for planning dynamics
    f_total = Fobs + ftar + fstoch;
66  %% 7. Compute angular velocity
    wrobot = f_total;
68  %% Compute linear velocity vector field
    % 8. Compute range of repulsion created by an obstacle
70  alpha = atan(C * Upot)/pi;
    % compute velocity contributions
72  c_obs = c_v_obs * (1/2 + alpha);
    c_tar = c_v_tar * (1/2 - alpha);
74  cv1 = -c_obs *( vrobot - Vobs) * exp(- (vrobot - Vobs)^2/(2*sigma_v ^2) );
    cv2 = -c_tar *( vrobot - Vtar) * exp(- (vrobot - Vtar)^2/(2*sigma_v ^2) );
76  % compute dynamic field
    g_v = cv1 + cv2;
78  %% 9. Set linear velocity
    if( alpha ) % if Thresholded potential is triggered
80      vrobot = vrobot + dt * g_v; % cm/s
    else % run at a constant speed
82      vrobot = 30;
    end
84  %% 10. Stop Criterion

86  %% view dynamics
    % visualization of the dynamics
88  phi_plot = (0:10:360)*pi/180;
    lphi = length(phi_plot);
90  % figure customization
    f = figure(1);
92  f.Name = 'Obstacle Avoidance dynamics';
    f.ToolBar = 'none'; f.MenuBar = 'figure';
```

```
94    clf;
      hold on
96    % for Obstacle Avoidance dynamics
      fobs_plot_all = 0;
98    upot_plot_all = 0;
      for j = 1:N
100       % 2.4. Compute f_obs_i
          aux = lambda_obs(j) .* exp(- (phi_plot - psi_obs(j)).^2/(2 * sigma(j) ^2));
102       fobs_plot = aux .* (phi_plot - psi_obs(j));
          % 2.5. Compute Lyapunov Function
104       k_i = lambda_obs(j) .* sigma(j)^2 / sqrt(exp(1));
          upot_plot = aux .* sigma(j)^2 - k_i;
106      % accumulate
          fobs_plot_all = fobs_plot_all + fobs_plot;
108       upot_plot_all = upot_plot_all + upot_plot;
      end
110   ftar_plot=-lambda_tar*sin(phi_plot -psi_tar); % target
      f_total_plot=ftar_plot+fobs_plot_all; % overall
112   alpha_plot = atan(C * upot_plot_all)/pi; % thresholded potential
      %plots
114   legStr = {'$f_{obs}$';'$f_{tar}$';'$f_{obs} + f_{tar}$';'$U(\phi)$'; '$\alpha(\phi)$'};
      p = plot(phi_plot * 180/pi, fobs_plot_all);
116   p.LineWidth = 2; p.LineStyle = '--';
      p = plot(phi_plot*180/pi,ftar_plot);
118   p.LineWidth = 2; p.LineStyle = '--';
      p = plot(phi_plot*180/pi,f_total_plot);
120   p.LineWidth = 3;
      p = plot(phi_plot*180/pi,upot_plot_all);
122   p.LineWidth = 2;
      p = plot(phi_plot*180/pi,alpha_plot);
124   p.LineWidth = 3;
      % axis
126   ax = gca; yl = ax.YLim; ax.XLim = [0, 360];
      ax.XTick = ax.XLim(1) : 45 : ax.XLim(2);
128   ax.Title.String = 'Behavioral dynamics';
      ax.XLabel.String = 'Heading direction - $\phi (^{\circ})$';
130   ax.YLabel.String = 'Magnitude';
      ax.Title.Interpreter = 'latex';
132   ax.YLabel.Interpreter = 'latex';
      ax.XLabel.Interpreter = 'latex';
134   ax.Title.FontSize = 14; ax.YLabel.FontSize = 14;
      % line
136   phi_robot_plot = phirobot;
```

```
    if(phirobot < 0)
138     phi_robot_plot = phirobot + 2*pi;
    end
140 l = line([ phi_robot_plot * 180/pi, phi_robot_plot * 180/pi], [yl(1), yl(2)]);
    l.Color = 'black';
142 % Legend
    lgd = legend(legStr);
144 lgd.Location = 'southeast';
    lgd.Interpreter = 'latex';
146 lgd.FontSize = 10;
    hold off
148 %%------------- END OF YOUR CODE -------------
    % set robot angular velocity (rad/s) and robot linear velocity (cm/s)
150 vehicle.set_velocity(wrobot, vrobot);
```

Listing 6.1: Generic implementation of the overall dynamics with control of driving speed (not tuned)

6.2 Simulations

Several simulations were performed to tune the multiple parameters and to assess the performance of the overall planning dynamics with control of driving speed.

6.2.1 Tuning

The parameters were then tuned, starting from the hierarchy of relaxation rates that ensures that the system relaxes to the attractors and that obstacle avoidance has precedence over the target, imposed as:

```
%% obstacle
2 tau_obs_min = 3.5 * dt; % flexible
  beta1 = 1/tau_obs_min; %  = lambda_obs_max = 1/tau_obs_min
4 %% target
  tau_tar = 20*tau_obs_min; % relaxation time for attractor
6 lambda_tar = 1/tau_tar; % attraction magnitude
  %% velocity
8 c_v_obs = 10 * beta1; % obstacle contribution to velocity
  c_v_tar = 100 * lambda_tar; % target contribution to velocity
```

Listing 6.2: Relaxation rates tuning

Next, the parameters directly related to the path velocity dynamics were tuned, as follows:

```
% Lyapunov
Rtar = 20; % target radius
d_min = Rrobot + Rtar + 5; % minimum distance to target
C = 100; % potential constant
d_obs = d_min; % minimum distance to obstacle
d_tar = d_min; % minimum distance to target
psi_max = pi/12; % maximum rate of fixed points shift
sigma_v = pi/6; % angular range for velocity
Vobs = d_obs * psi_max; % maximum velocity for obstacle avoidance behavior
Vtar = d_tar * psi_max; % maximum velocity for target acquisition behavior
```

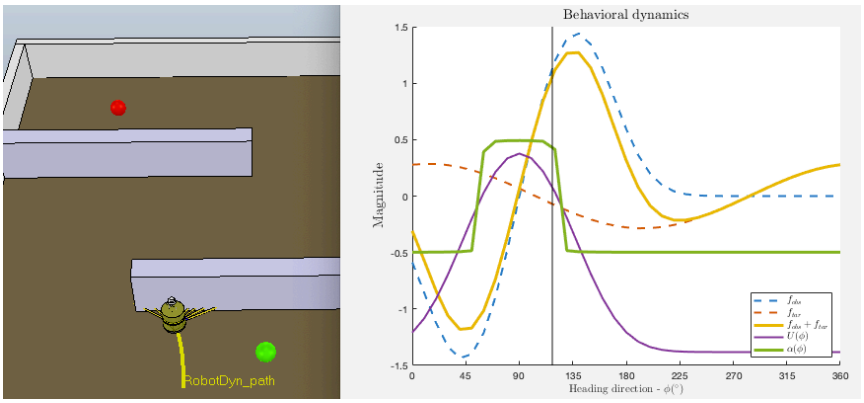
Listing 6.3: Path velocity parameters tuning

The potential constant, C, was defined high enough to obtain a sharp sigmoid, required for the transition behavior. The distance to obstacle and target were considered as the minimum one, corresponding to the robot's size. The maximum rate of fixed points shift,$\dot{\psi}_{max}$, was set to a relatively small value, so the system can closely follow the attractors. The angular range for velocity, σ_v, defines the range where the obstacles presence must be accounted for path velocity dynamics,and this was set to two sensor sectors.

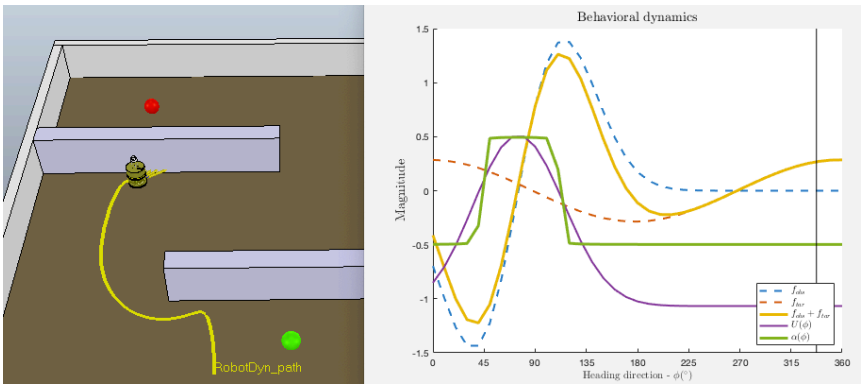
6.2.2 S scenario

Then, the robot's planning dynamics was simulated using the **MobileRobotDyn_Tar_Obs_S.ttt** scenario (see Fig. 6.2 and Video ./videos/lyapunov-s.mp4). The robot moves initially with a constant, predefined, speed, with a slight tilt due to the orientation of the target, ψ_{tar}. When the robot approaches the wall, the obstacles are detected in a significant way (Fig. 6.2a), with the overall dynamics establishing a repeller on the 90 degrees heading direction. Consequently, the obstacles potential is positive in the vicinity of the repeller, i.e. $U(\phi) > 0$, and, thus, the thresholded potential is also positive, i.e. $\alpha(\phi) > 0$, with the defined angular range σ_v. Hence, the robot starts to turn left – $\phi > 90°$ – with the path velocity defined by Eq. (6.3), affected by the sigmoidal function and the associated parameters.

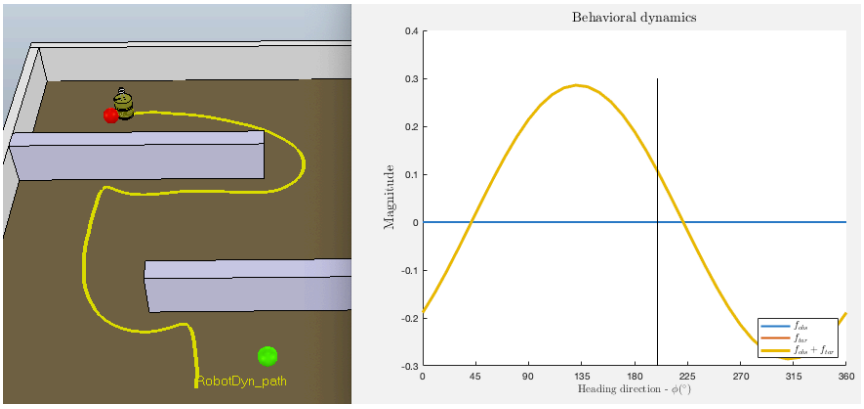
Fig. 6.2b shows a similar situation, but now the robot turns right – $\phi \approx -45°$. Lastly, Fig. 6.2c illustrates the final state of the simulation when the robot reaches the target, and only its contribution is present. There is slight mismatch between the heading direction and the closest attractor due to the imposed path velocity dynamics. Comparing this simulation to the nonlinear dynamics for heading direction with linear path velocity dynamics (Fig. 4.6), it can be observed a more 'clean' and shortest path, although with greater cornering radius, due to the increased average speed.



(a)



(b)



(c)

Figure 6.2: Planning dynamics with control of driving speed: Scenario S

6.2.3 Tar-Obs Scenario

Then, the robot's planning dynamics was simulated using the **MobileRobotDyn_Tar_Obs.ttt** scenario with a 50 cm gap between obstacles (see Fig. 6.3 and Video ./videos/lyapunov-tar-obs.mp4).

The robot moves forward, towards the target, until it starts to detect the obstacles (Fig. 6.3a). Although

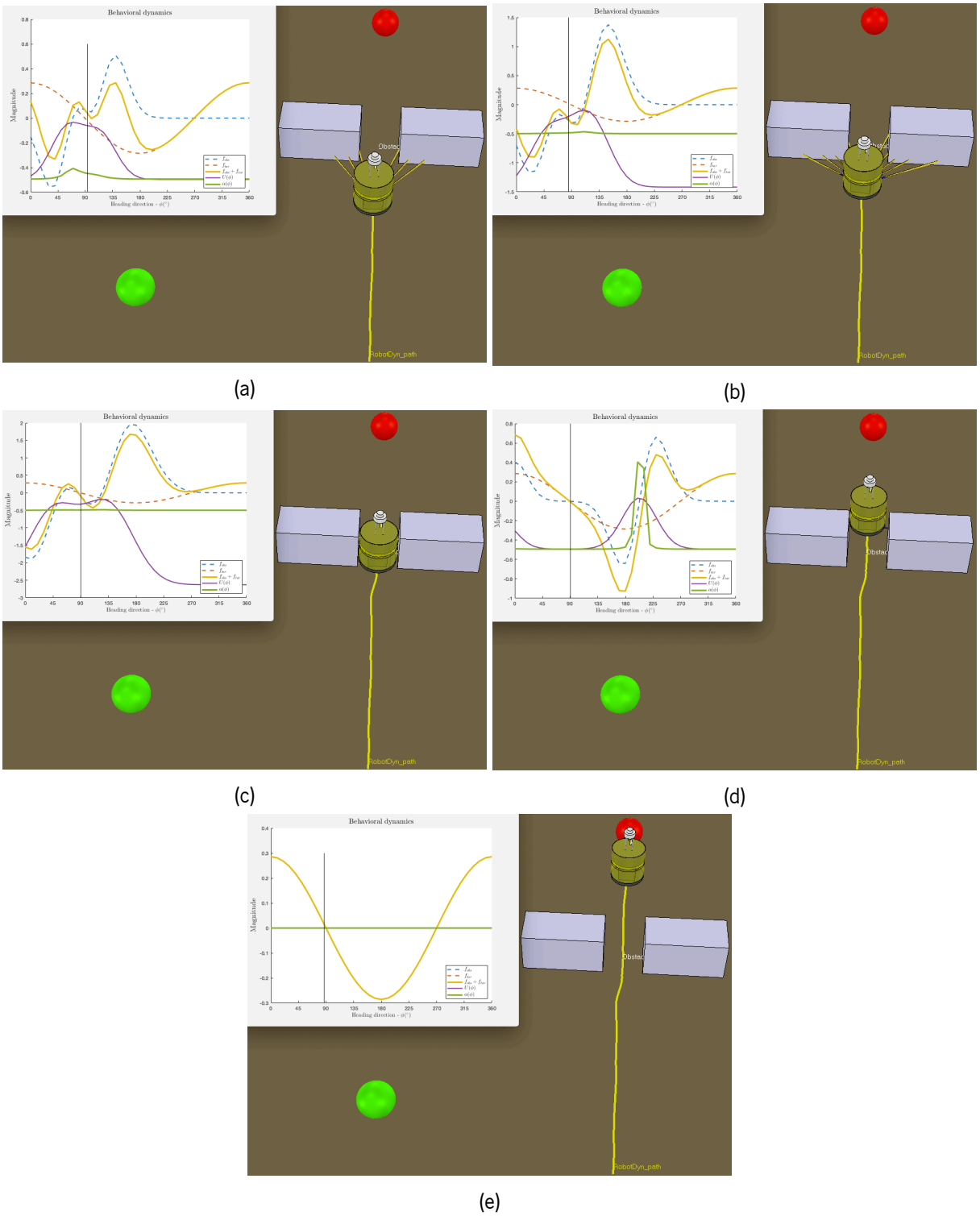


Figure 6.3: Planning dynamics with control of driving speed: Scenario Tar-Obs

the potential function is not null, it is negative as expected, as the target acquisition behavior is prevalent — an attractor is established in the heading direction. Thus, only the target contributes to the path velocity dynamics. Fig. 6.3b and Fig. 6.3c illustrate a similar behavior, although with slight variations in the potential function, as the robot moves in between the obstacles. Fig. 6.3d showcases the only brief moment when the thresholded potential is positive, i.e. $\alpha(\phi) > 0$, as the robot escapes the obstacles wall and the obstacle avoidance prevails over the target acquisition behavior for the path velocity dynamics. Lastly, Fig. 6.3e shows the moment the robot reaches the target with an attractor in the heading direction as expected. This simulation showcases a critical scenario, where the robot must pass between the obstacles at a minimum distance, yet, at a adequate velocity.

6.3 Discussion

The control of driving speed is critical for maintaing the stability of the planning dynamics of the robot, as it limits the rate of fixed points shifts, enabling the system to track the attractor as it shifts.

The shifting of fixed points for the planning dynamics can stem from robot movement through the environment and associated sensory information changes or due to environmental changes (obstacles moving in the world), causing attractors and repellers to change. To keep the system stable, i.e. in or near an attractor at all times, the rate of such shifts must be limited to permit the track the attractor as it shifts. One way of accomplishing this is by controlling the path velocity of the vehicle, as the rate of fixed points shift is determined by the relative velocity of the robot with respect to its environment [1].

In this chapter, a more adequate dynamic system for path velocity was established, as the sum of obstacles and target contributions, where one of the components dominates at all times. A systematic way to construct a function that indicates if obstacles contributions are present, is to integrate force-lets, from which a potential function of the obstacle avoidance dynamics results. If the potential is positive a repeller is established for the planning dynamics, and conversely, if negative an attractor arises. A convenient way to transform the potential levels to the strengths of the two contributions to the velocity control is through the use of a sigmoidal threshold function, defining the angular range the obstacles contribution is noticeable.

Then, the parameters were tuned considering the hierarchy of relaxation rates — which ensures that the system relaxes to the attractors and that obstacle avoidance has precedence over the target — and some rules of thumb for the remaining path velocity parameters.

Finally, two scenarios were simulated — S and Tar-Obs — to assess the performance of the overall dynamics. It was shown that, although the average velocity was increased, the planning dynamics remains robust, suggesting a performance improvement in the overall dynamics.

7 Conclusion

In this chapter are outlined the conclusions and the prospect for future work regarding the autonomous navigation of robots.

7.1 Conclusions

The autonomous navigation of robots can be described as a collision-free path towards a target and successfully formulated and implemented using dynamic systems. Amongst the class of the dynamic systems, the nonlinear type arises as the most useful due to variations of qualitative behavior — number and or/or stability of the fixed points of the dynamic system.

Then, the behaviors were analyzed in isolation and in integration, namely target acquisition and obstacle avoidance, strongly supported by a previous analytical study of the corresponding dynamic system. This analytical study, based on the qualitative theory of dynamic systems, consisted in the determination of the fixed points and its stability, the corresponding phase portraits, i.e., possible evolution of the system's state, and the bifurcation diagram, yielding the bifurcation points as a function of the parameters of the system.

Target Acquisition: An autonomous mobile robot must be able to navigate to a target position. The target acquisition behavior can be yielded by dynamic systems for the heading direction and path velocity (Eq. (2.1) and Eq. (2.2)). The heading direction dynamics consists of a sinusoidal function (Eq. (2.3)), as it is required the heading direction converges to the target independently of its initial state, yielding an attractor in the heading direction and a repeller in the opposite one for faster convergence.

The implementation of the dynamic systems was performed using Forward Euler's method in Matlab, and used to command a simulation in CoppeliaSim environment. Several scenarios were simulated for different velocities functions, and was observed that a dynamic system for path velocity is required for adequate velocity control. Alongside with these simulations, the system's parameters were tuned, namely the time constants of the system.

Lastly, a linear dynamic system for heading direction was considered and compared to the nonlinear

one. It was observed the nonlinear dynamic system is better suited for the target acquisition behavior of the robot, as it provides smoother paths — narrower angular velocity range.

Obstacle avoidance: While moving to a target, a robot must also avoid obstacles that may appear — obstacle avoidance. To avoid obstacles, the robot must firstly detect them, in this case, using infrared radiation sensors. The strategy adopted consisted in assuming that each sensor i specifies a virtual obstacle in the direction $\psi_{obs,i}$ if an obstruction is detected in that direction, modelled by a repulsive force centered in the direction the respective sensor points out, $f_{obs,i}$ (Eq. (3.1)). As each sensor is mounted in a fixed direction θ_i in respect to the frontal direction of the robot, the calibration of the system is unnecessary.

The obstacle avoidance dynamics establishes a repeller in the virtual obstacle direction with variable strength, as a function of the distance to obstacles (Eq. (3.2)) — high for small distances and low for high distances — and where the angular range over which the force-let exerts its repulsive effect should be limited (Eq. (3.3)). Gaussian white noise is also added to the dynamics to enable the escape from the repeller in a finite time, in case this is the initial condition.

The implementation was performed in Matlab using Forward Euler's method. Several simulations were then performed to evaluate the impact of parameters' variations, namely the maximum strenght of repulsion, β_1, the decay rate of the repulsion force with the distance increase, β_2, and noise magnitude Q. β_1 relates to the minimum time constant of the obstacle avoidance dynamics $\tau_{obs,i}$ (Eq. (3.12)), and for the simulations performed $\beta_1 = 3.5\Delta t$. It should be noted, however, that the value of this parameter is dependent on the rate of fixed points shift. As for the parameter β_2, for increasing values the decay rate diminishes, maintaining the repulsive effect significantly for a wider distance range — the robot starts to rotate farther away from the obstacles. Conversely, for decreasing values of β_2, the decay rate increases, maintaining the repulsive effect significantly for a narrow distance range — the robot starts to rotate closer to the obstacles. The addition of noise is important, but should be limited in magnitude to avoid jitter: it should be sufficient enough to guarantee the escape from the repellers within a time limit, in case the system is initially placed there. Additionally, noise induces different qualitative behaviors (turn left/right) due to stochastic effect. It was also noted that noise may not always be required, as the system may present slight bias in sensors readings.

The simulations with different gap between obstacles showed a the heading direction dynamics exhibits different qualitative behaviors — the stability of the fixed points varies — starting from gap = 50 cm. This corresponds to a bifurcation point, representing the distance below which the robot (with diameter 45 cm) cannot pass between the two obstacles. For gap < 50 cm, the planning dynamics has an reppeller at the heading direction $\phi = \pi/2$, and for gap > 50 this reppeller becomes asymptotically stable (i.e., an attractor).

Lastly, a simulation was performed in an environment with multiple obstacles, with the robot successfully and adequately avoiding all of them.

Integration: Obstacle avoidance and Target acquisition: Next, the obstacle avoidance and target acquisition behaviors were integrated, considering two approaches for the latter: nonlinear or linear dynamic system. For both approaches the obstacle avoidance behavior must take precedence over the target acquisition to prevent the robot from hitting an obstacle while attempting to move to the target. This is guaranteed by imposing greater magnitude for repulsive component than for attractive one, i.e. $\lambda_{obs} \gg \lambda_{tar}$, and consequently, $\tau_{obs} \ll \tau_{tar}$. After analyzing the robot's behavior for both nonlinear and linear dynamics, it is was possible to compare them. Mathematically, the fact that the nonlinear function is a sinusoidal function, yielding always two fixed points — one attractor, and in the opposite direction a repeller — whereas the linear function only contributes with one attractor to the vector field. The repeller in the opposite direction to the target reinforces the heading direction that must be taken by the robot so that it reaches the target. In the previous experiments, as well by Section 2.4.5, it is concluded by experimentation too, that the nonlinear system provides smoother paths at narrower angular velocity range. Thus, it can be concluded that the nonlinear dynamic system is better for the target acquisition behavior of the robot.

Control of driving speed The stability of the planning dynamics is affected by the rate of fixed points shift, as to keep the system stable, i.e. in or near an attractor at all times, the rate of such shifts must be limited to permit the track the attractor as it shifts. One way of accomplishing this is by controlling the path velocity of the vehicle, as the rate of fixed points shift is determined by the relative velocity of the robot with respect to its environment [1].

The shifting of fixed points for the planning dynamics can stem from robot movement through the environment and associated sensory information changes or due to environmental changes (obstacles moving in the world), causing attractors and repellers to change.

Thus, a more adequate dynamic system for path velocity was established, as the sum of obstacles and target contributions, where one of the components dominates at all times. A systematic way to construct a function that indicates if obstacles contributions are present, is to integrate force-lets, from which a potential function of the obstacle avoidance dynamics results. If the potential is positive a repeller is established for the planning dynamics, and conversely, if negative an attractor arises. A convenient way to transform the potential levels to the strengths of the two contributions to the velocity control is through the use of a sigmoidal threshold function, defining the angular range the obstacles contribution is noticeable.

Then, the parameters were tuned considering the hierarchy of relaxation rates — which ensures that the system relaxes to the attractors and that obstacle avoidance has precedence over the target — and some

rules of thumb for the remaining path velocity parameters.

Finally, two scenarios were simulated — S and Tar-Obs — to assess the performance of the overall dynamics. It was shown that, although the average velocity was increased, the planning dynamics remains robust, suggesting a performance improvement in the overall dynamics.

Final remarks: Dynamic systems can be used to successfully implement the autonomous mobile navigation of robots in a robust way, as the system follows closely the asymptotically stable states, i.e. the attractors, yielding it robust to noise and external disturbances. Specifically, nonlinear dynamic systems provide a richness of qualitative behaviors as required for the highly dynamic conditions imposed by the surroundings. Although more complex than the linear counterparts, the qualitative theory of dynamic systems provides a convenient framework for a relatively easy analysis of nonlinear dynamic systems.

The autonomous navigation of robots can be described as a collision-free path towards a target, thus requiring both obstacle avoidance and target acquisition behaviors for the planning dynamics. For the former, a repeller is established in the direction of the virtual obstacles detected by the robot's sensor; for the latter, an attractor is established in the direction of the target and a repeller in the opposite direction for faster convergence of the desired heading direction. In the integration of both behaviors the hierarchy of time constants must be observed, assuring obstacle avoidance precedence over target acquisition and in a adequate form, related to the computing time of the system.

To keep the planning dynamics stable, i.e. in or near an attractor at all times, the rate of such shifts must be limited to permit the track the attractor as it shifts. One way of accomplishing this is by controlling the path velocity of the vehicle, as the rate of fixed points shift is determined by the relative velocity of the robot with respect to its environment [1]. Thus, a more adequate dynamic system for path velocity can be established using obstacles and target contributions to the dynamic through a thresholded potential function and a convenient tuning of the system.

7.2 Prospect for Future Work

In the foreseeable future, the deployment of the planning and path velocity dynamics to hardware could be performed, analyzing the navigation behavior of the robot in real world scenarios. Additionally, the detection of targets could be performed by the robot using different techniques such as sound detection or computer vision.

Another interesting feature is to incorporate artificial intelligence to the robot, enabling it to recall the most relevant target and to forget the others, making it even more adaptable to the highly dynamic conditions of the surroundings.

Bibliography

[1] Estela Bicho. Dynamic Approach to Behavior-Based Robotics: design, specification, analysis, simulation and implementation. Shaker Verlag, 2000.

[2] Estela Bicho and Gregor Schöner. The dynamic approach to autonomous robotics demonstrated on a low-level vehicle platform. Robotics and autonomous systems, 21(1):23–35, 1997.