Robot Navigation in Presence of People

Konstantinos Christopoulos

Supervisor: Konstantinos Vlachos

Department of Computer Science & Engineering

Ioannina, February 2025

Introduction

Problem

Safe navigation in environments with people and obstacles.

Importance

Pedestrian avoidance is crucial in robotics, due to the **unpredictable human movement.**

Objective

Developing a pedestrian-aware navigation framework in order to avoid collisions.

System Overview

Jackal Robot

Equipped with a Velodyne LiDAR, that emitting laser pulses

ROS

Robot Operating System for software development

Navigation Process

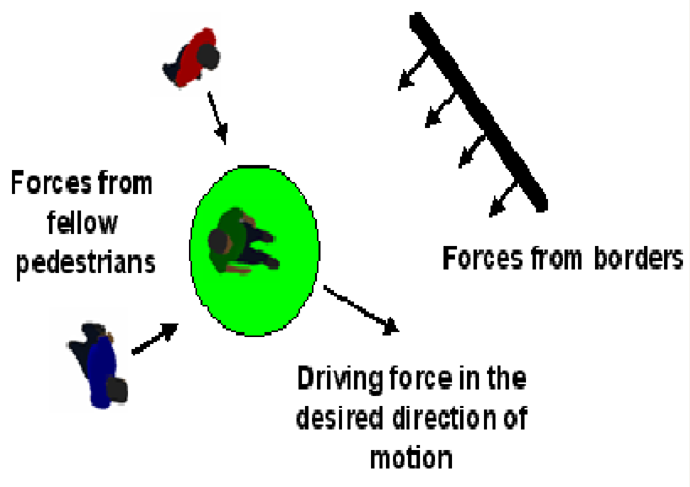
1. Pedestrian & obstacles detection
2. Pedestrian tracking
3. Pedestrian & obstacles avoidance

Data Flow

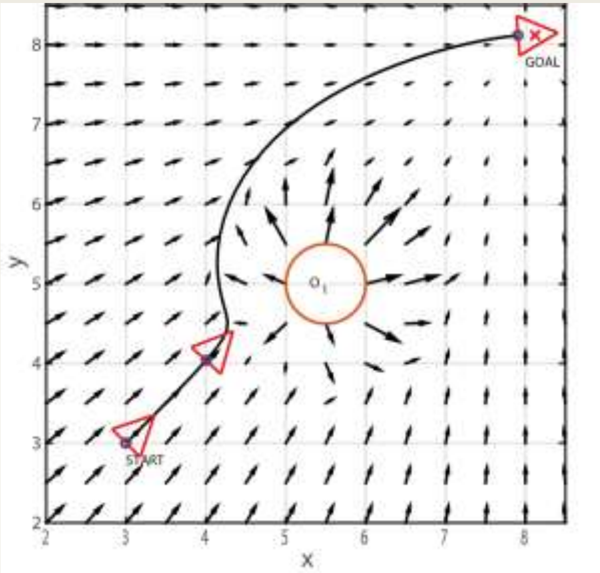
ROS nodes and topics for data exchange

Jackal Robot in Lab

Approach



Social Force Model



Artificial Potential Field

Social Force Model (SFM)

Utilizing SFM for pedestrian motion

Pedestrian Detection & Tracking

LiDAR-based detection and tracking of the pedestrian

Potential Field Navigation

Implementing APF motion algorithm for robot


Robot in Crowded Environment

Social Force Model (SFM)

The Social Force Model (SFM) is a widely used approach for modeling pedestrian motion. This model represents the pedestrian's motion as the result of various social forces, including:

1 Desired Force

Represents a person’s desire to move in a particular direction

2 Obstacle Force

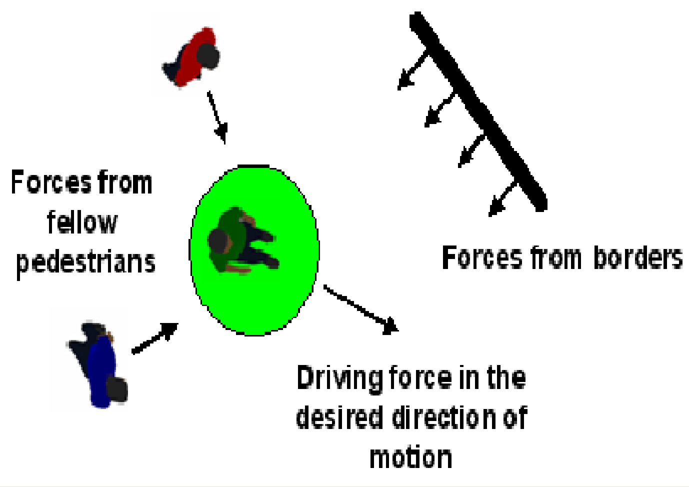
A repulsive force to avoid conflicts with obstacles in the environment

3 Social Force

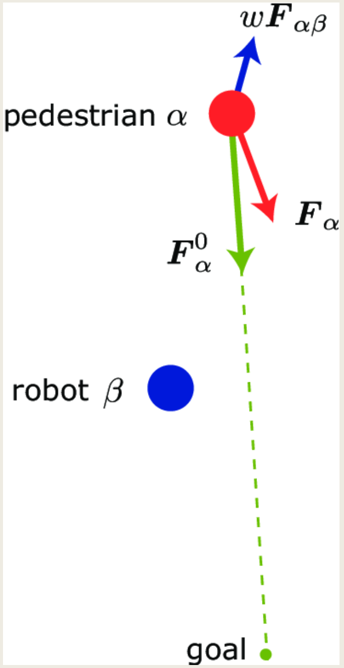
A force caused by the interaction between pedestrians that causes them to avoid each other to avoid conflict

4 Robot Force

A repulsive force to avoid conflicts with the robot in the environment



Social Force Model



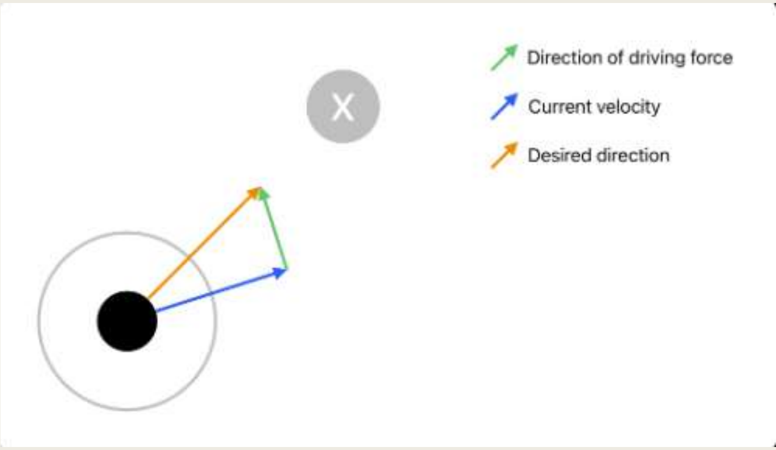
Robot Force

Social Force Model (SFM)

The Social Force Model (SFM) is a widely used approach for modeling pedestrian motion. This model represents the pedestrian's motion as the result of various social forces, including:

1 Desired Force

Represents a person's desire to move in a particular direction.
Each pedestrian aims to move towards a particular destination.



Desired Force

$$e_a(t) := \frac{r_a^k - r_a(t)}{\|r_a^k - r_a(t)\|}$$

$$\overrightarrow{F_a^0}(\vec{u_a}, \vec{u_a}\vec{e_a}) := \frac{1}{\tau_a}(u_a^0\vec{e_a} - \vec{u_a})$$

- pedestrian α
- r_a^k: the next target point
- r_a^0: the final destination
- $r_a(t)$: current position
- $e_a(t)$: desired direction

- $\vec{u_a}$: actual speed of the pedestrian
- u_a^0: desired speed
- τ_a: relaxation time (how quickly the pedestrian adjust his current speed)
- $\overrightarrow{F_a^0}(\vec{u_a}, \vec{u_a}\vec{e_a})$: desired force

Social Force Model (SFM)

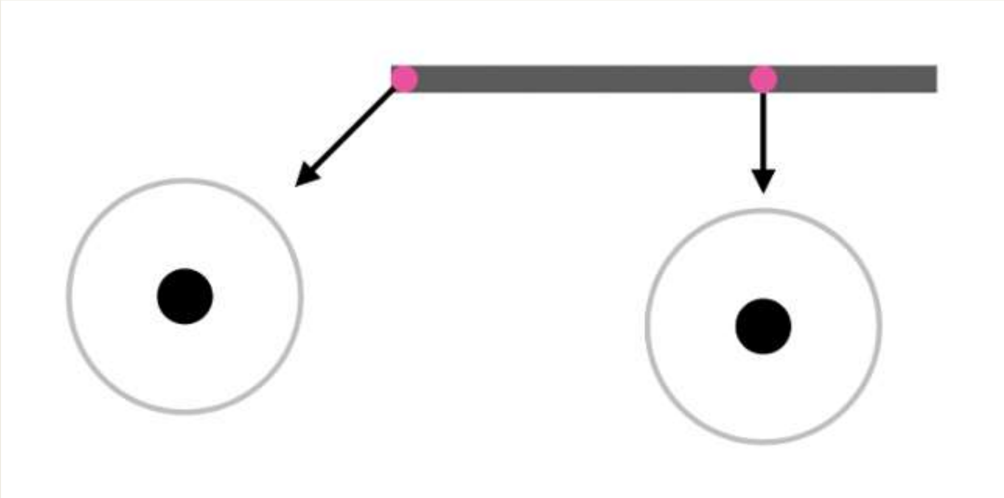
The Social Force Model (SFM) is a widely used approach for modeling pedestrian motion. This model represents the pedestrian's motion as the result of various social forces, including:

2 Obstacle Force

A repulsive force to avoid conflicts with obstacles in the environment.
The pedestrian α wants to keep a certain distance from obstacles.

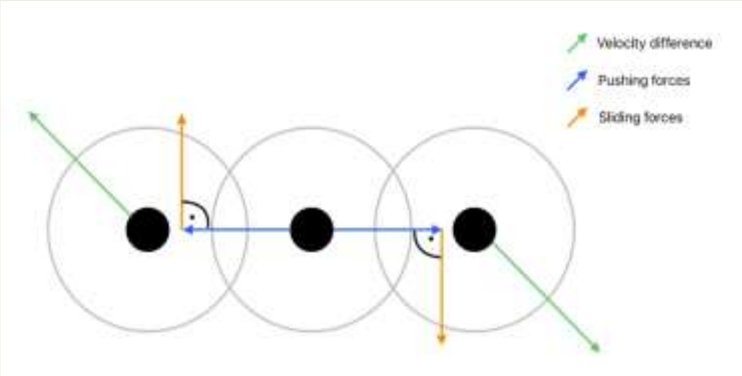
$$F_{aB} = \frac{1}{B} e^{-\frac{\|\vec{r_{aB}}\|}{B}} \frac{\vec{r_{aB}}}{\|\vec{r_{aB}}\|}$$

- $\vec{r_{aB}} = r_a - r_B$: vector from the pedestrian α to the closest point to an obstacle B
- B: parameter that controls the "width" of the repulsion
- F_{aB}: Obstacle force



Obstacle Force

Social Force Model (SFM)



Social Force

The Social Force Model (SFM) is a widely used approach for modeling pedestrian motion. This model represents the pedestrian's motion as the result of various social forces, including:

3 Social Force

A force caused by the interaction between pedestrians (α and β) that causes them to avoid each other to avoid conflict.

$$f_v(d,\theta) = -Ae^{-\frac{d}{B}-(n'B\theta)^2}$$

$$f_\theta(d,\theta) = -AKe^{-\frac{d}{B}-(nB\theta)^2}$$

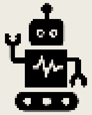
$$F_{\alpha\beta}(d,\theta) = -Ae^{-\frac{d}{B}}[e^{-(n'B\theta)^2}t_{\alpha\beta} + e^{-(nB\theta)^2}n_{\alpha\beta}]$$

- f_v: the deceleration along the interaction direction $t_{\alpha\beta}$
- f_θ: the directional changes along $n_{\alpha\beta}$
- $\theta_{\alpha\beta}$: the angle formed between the direction of interaction $t_{\alpha\beta}$ and the vector pointing from pedestrian α to β
- $n_{\alpha\beta}$: the normalized vector of $t_{\alpha\beta}$ oriented to the left.
- $t_{\alpha\beta} = \frac{D_{\alpha\beta}}{\|D_{\alpha\beta}\|}$
- $e_{\alpha\beta} = \frac{x_\beta - x_\alpha}{\|x_\beta - x_\alpha\|}$
- x_α, x_β: location of pedestrian α, β
- $D_{\alpha\beta} = \lambda(u_\alpha - u_\beta) + e_{\alpha\beta}$
- u_α, u_β: velocity of pedestrian α, β
- λ: relative importance of the two directions
- d: distance between 2 pedestrians
- n, n', B, K, Λ: parameters
- $F_{\alpha\beta}(d,\theta)$: Social force

- f_v: decreases exponentially as the distance between 2 pedestrians increases (d↑).
- f_θ: focus more on the directional changes.

Social Force Model (SFM)

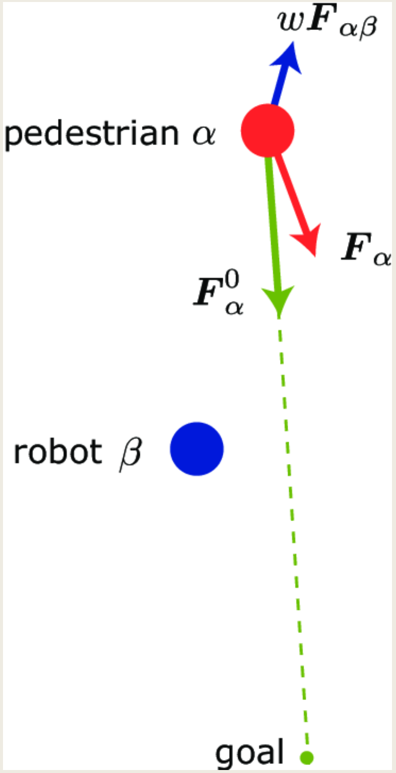
The Social Force Model (SFM) is a widely used approach for modeling pedestrian motion. This model represents the pedestrian's motion as the result of various social forces, including:

4 Robot Force 

A repulsive force to avoid conflicts with the robot in the environment

$$F_{robot} = -e^{-\frac{\|r_{robot}-r_a\|}{\sigma_{robot}}} \frac{r_{robot}-r_a}{\|r_{robot}-r_a\|}$$

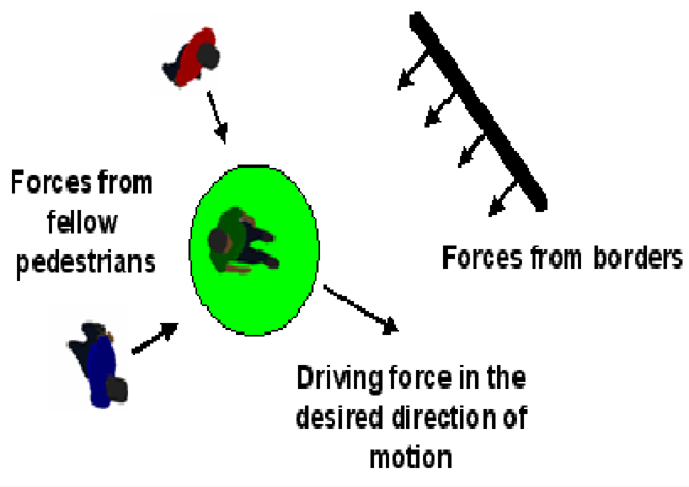
- r_{robot}: position of the robot
- r_a: position of the pedestrian
- σ_{robot} parameter (indicates how fast the F_{robot} decreases with distance)
- F_{robot}: Robot force



Robot Force

Social Force Model (SFM)

The Social Force Model (SFM) is a widely used approach for modeling pedestrian motion. This model represents the pedestrian's motion as the result of various social forces, including:



Social Force Model

Total Force

All the previous effects influence a pedestrian's decision at the same moment

$$F_{total} = F_a^0(t) + F_{aB}(t) + F_{a\beta}(t) + F_{robot}$$

- $F_a^0(t)$: Desired force
- $F_{\alpha B}(t)$: Obstacle force
- $F_{\alpha\beta}(t)$: Social force
- F_{robot}: Robot force
- F_{total}: Total force

Velocities & Positions Updates

According to the 2nd law of Newton and using Euler method.

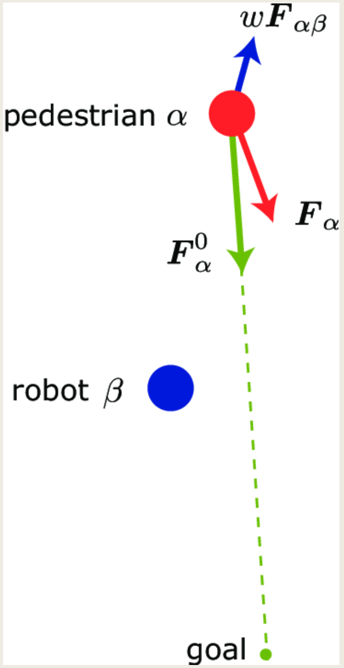
$$\vec{F} = m\vec{a} \Rightarrow \frac{\partial u_a}{\partial t} = \vec{a} = \vec{F_{total}} \Rightarrow$$

$$u_a(t+1) = u_a(t) + stepSize * \vec{a}$$

$$r_a(t+1) = r_a(t) + stepSize * u_a(t+1)$$

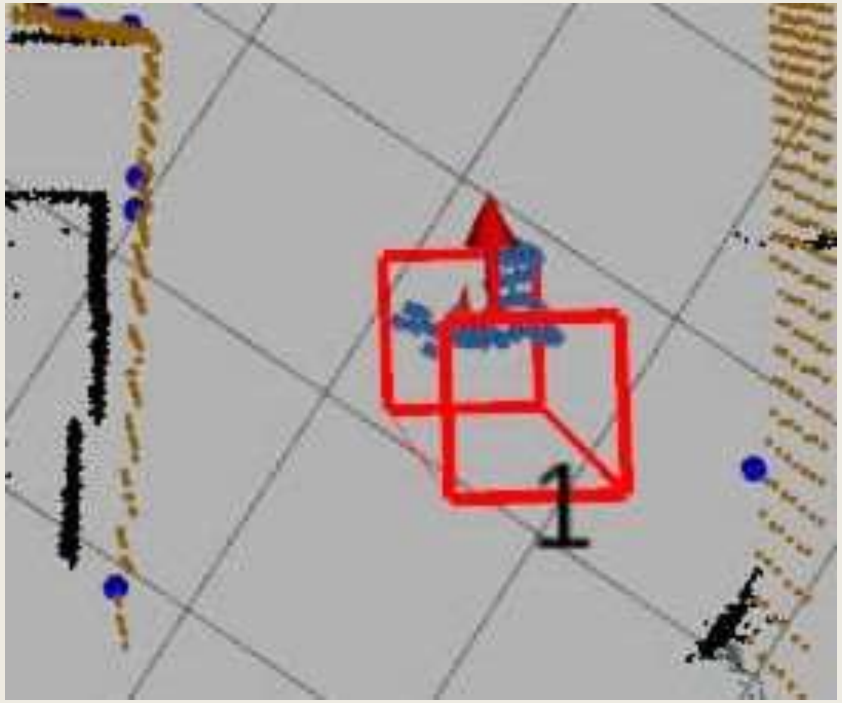
We consider
m = 1kg

- $u_a(t), u_a(t+1)$: current and updated velocity of the pedestrian
- $r_a(t), r_a(t+1)$: current and updated position of the pedestrian



Robot Force

Pedestrian Detection & Tracking



LiDAR-based Detection

Utilizing LiDAR data to identify pedestrians and obstacles

Motion Filtering

Filtering based on motion information

DBSCAN Clustering

Clustering algorithm for pedestrian identification

Kalman Filtering

Predicting pedestrian movement using Kalman filters

Pedestrian Detection & Tracking

Motion-based Filtering

Filtering static and dynamic points based on motion data information.

- We compare the current LiDAR scan to an earlier "**initial**" snapshot. Points that have **appeared** or **shifted significantly** are treated as **dynamic.**

Min Distance** for **data association

For each current point cloud point

$$d(\mathbf{p}_{curr}) = \min_{\mathbf{p}_{init} \in C_{init}} \|\mathbf{p}_{curr} - \mathbf{p}_{init}\|$$

- $p_{curr} = (x_{curr}, y_{curr})$: LiDAR data
- C_{init}: set of all "initial" points

$d(\mathbf{p}_{curr}) > \delta_{motion}$ → Dynamic Points

$d < threshold$ → Static Points

$\delta_{motion} = 0.2m$

! **Same method** for filtering **static** and **dynamic** points

Pedestrian Detection & Tracking

DBSCAN Clustering (Density-Based Spatial Clustering of Applications with Noise)

Once static obstacles are removed, we use DBSCAN Clustering algorithm,

to cluster post-filtered ‘dynamic’ points, in order to identify pedestrian.

- Identifies clusters of arbitrary size.
- Handles noise reliably

Dynamic Points

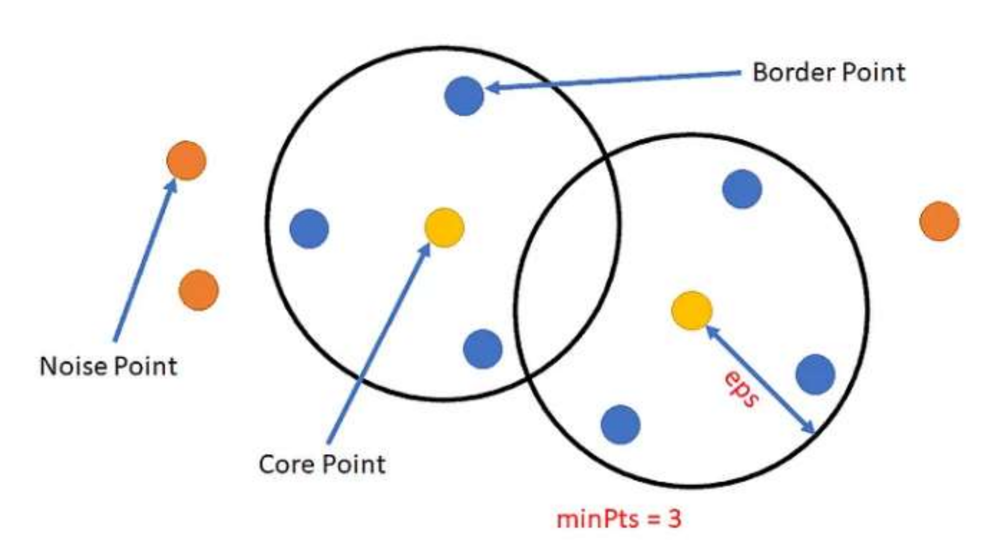
⬇

DBSCAN Clustering

minPts = 40
ε-radius = 0.6m

⬇

Pedestrian Clusters



Pedestrian Detection & Tracking

Kalman Filters

Predicting pedestrian movement using Kalman filters. It operates in a *prediction-update cycle* to obtain estimates of unknown variables that are more accurate than those on a single measurement.

1 Predict Step

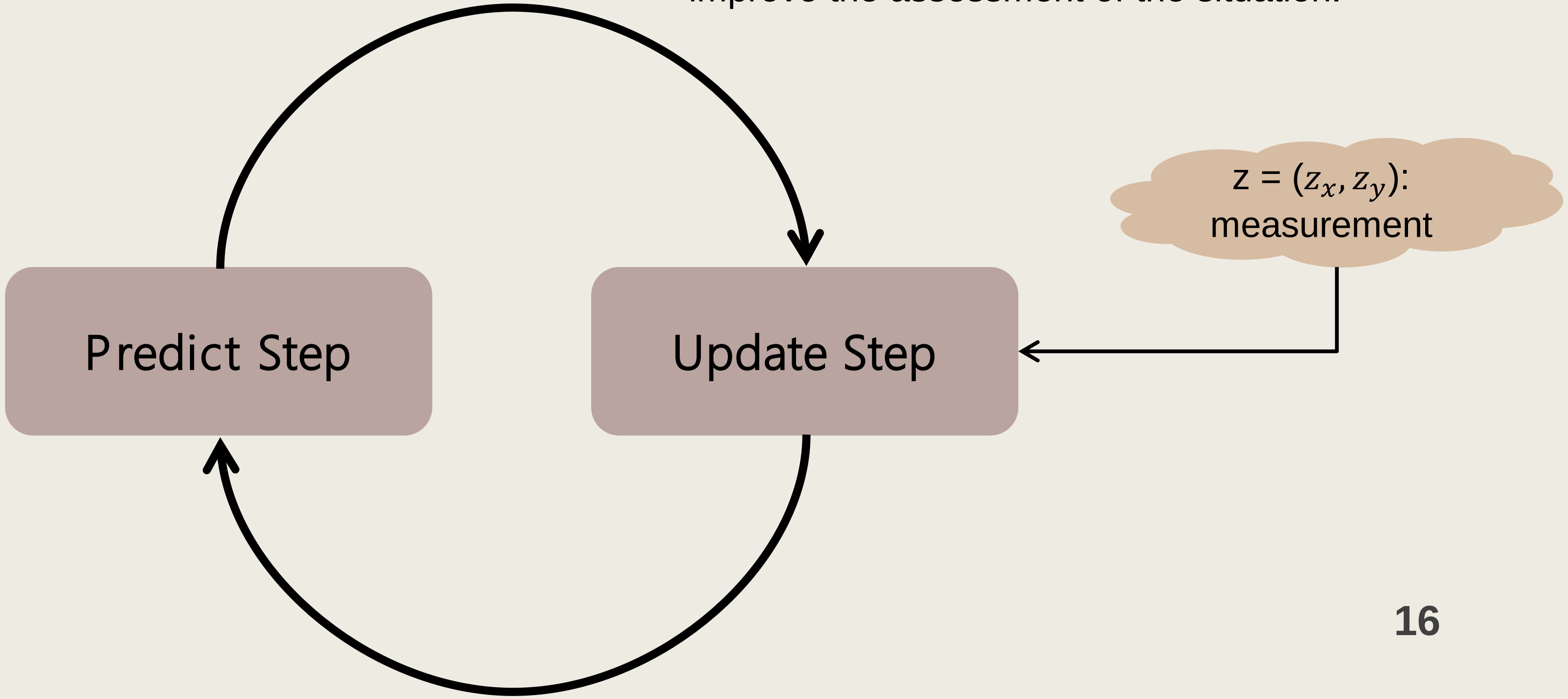
We predict the next situation and uncertainty.

2 Update Step

We incorporate the new measurement to improve the assessment of the situation.

State model:
- x, y: position
- u_x, u_y: velocity

$$\vec{X} = \begin{bmatrix} x & y & v_x & v_y \end{bmatrix}^T$$



Avoidance Strategy

1

Pedestrian Detection

Robot detects approaching pedestrian

2

Selection the Side of Movement

Robot selects side (right or left) in order to move

3

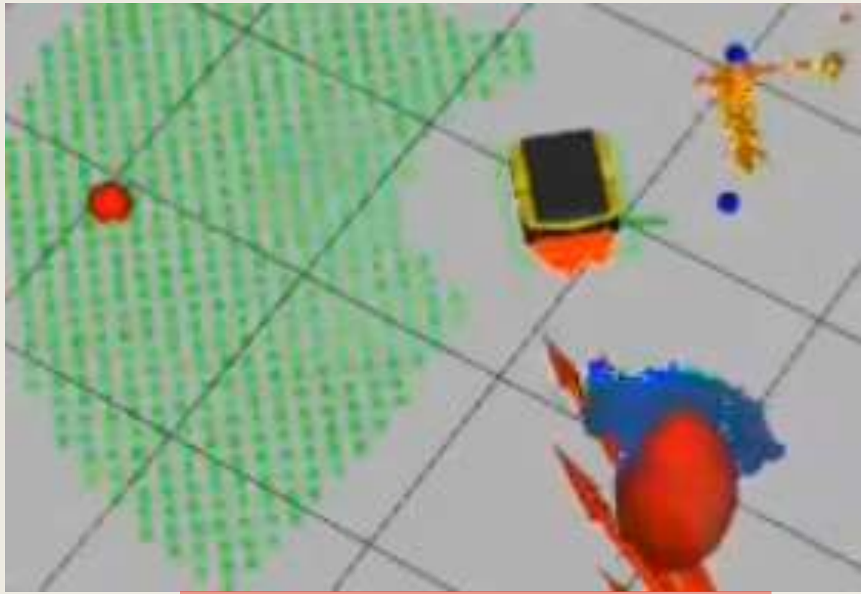
Determination of Best Free Point

Robot finds the best free point of the specified side

4

Potential Field Navigation

Balancing attractive and repulsive forces and moves to the best free point



- green circles: potential points
- red circle: optimal point
- red ellipse: pedestrian

Avoidance Strategy

1

Identification of Approaching Pedestrian

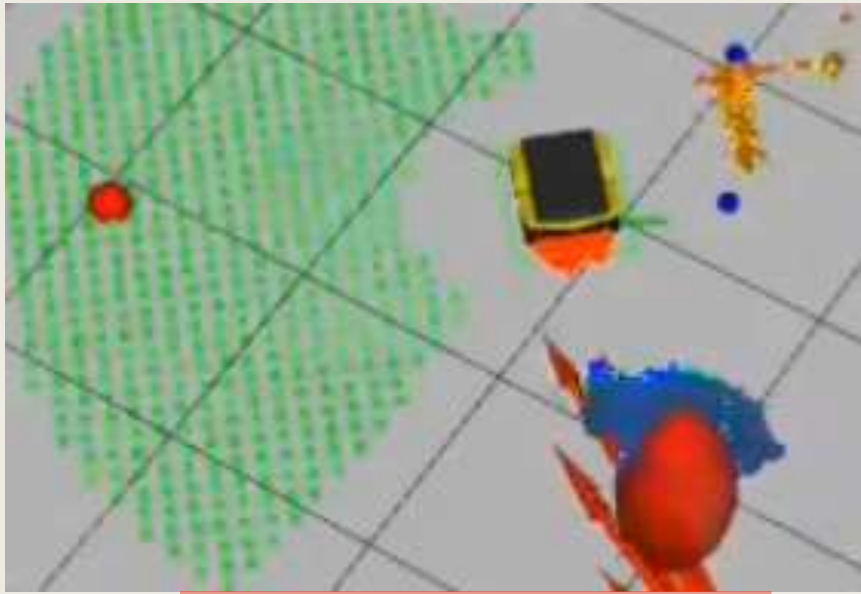
Robot detects approaching pedestrian.

$$r = \mathbf{p}_{robot} - \mathbf{p}_{ped}$$

$$\theta = \cos^{-1}\left(\frac{\mathbf{v}_{ped} \cdot \mathbf{r}}{\|\mathbf{v}_{ped}\|\|\mathbf{r}\|}\right)$$

- p_{robot}, p_{ped}: position of the robot and the pedestrian
- v_{ped}: velocity of the pedestrian from Kalman Filter
- r: vector from pedestrian to the robot
- θ: angle between the velocity vector and the pedestrian-to-robot vector

$\|r\| < 2.5$ m ✚ θ < 30°



- green circles: potential points
- red circle: optimal point
- red ellipse: pedestrian

2

Analyzing robot's surroundings

Robot analyze the LiDAR data to see which side (right or left) offers a clearer path to move.

Reference "**forward** vector" r.

Left/Right vector: r ±90°.

Data counts: LiDAR data points **p** (±45° cones around left/right vector and distance < 2.5m).

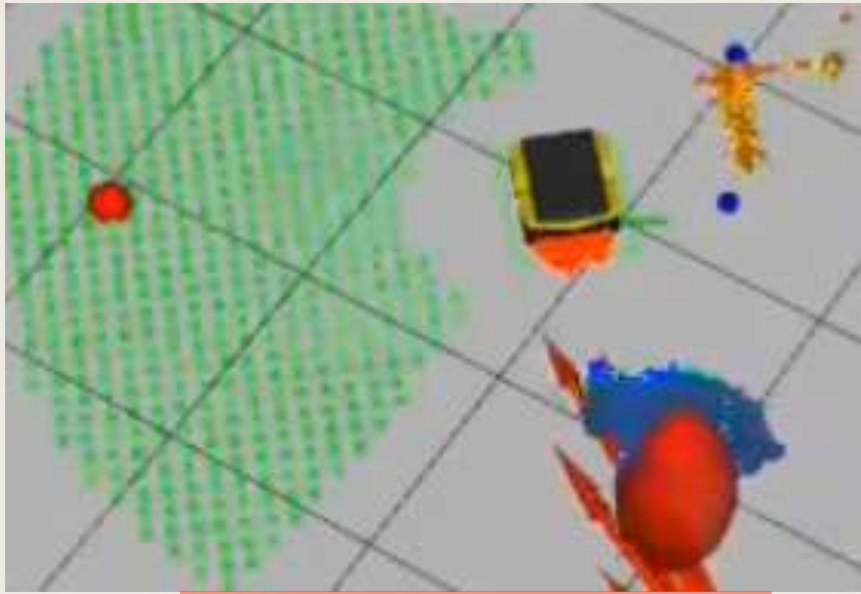
➡ **Decision**: choose side with fewer data returns.

Avoidance Strategy

3

Determination of Best Free Point

Robot finds the best free point of the specified side. We select the optimal point that maximize the distance from obstacle.



- green circles: potential points
- red circle: optimal point
- red ellipse: pedestrian

Dimensions ($max_{distance}$ = 2.5m)

Generate a grid of potential points (rectangular in robot's frame).

- $\mathrm{x}_{min} = \mathrm{y}_{min} = -\mathrm{max}_{distance}$
- $\mathrm{x}_{max} = \mathrm{y}_{max} = \mathrm{max}_{distance}$

Filter candidates points by angle (<30°).

$$\theta = \cos^{-1}\left(\frac{\vec{\mathbf{p}_i} \cdot \vec{\mathbf{movement}}_{direction}}{\|\vec{\mathbf{p}_i}\| \|\vec{\mathbf{movement}}_{direction}\|}\right)$$

For each candidate point compute minimum distance from every obstacle.

$$d(p_i) = \min_{\mathbf{o} \in Obs} \|\mathbf{p}_i - \mathbf{o}\|$$

- $\vec{\mathbf{p}_i}$: vector from robot to potential point
- $\vec{\mathbf{movement}}_{direction}$: vector perpendicular to the forward "r" vector on the selected side

Largest minimum distance to obstacles

Select the Safest Position.

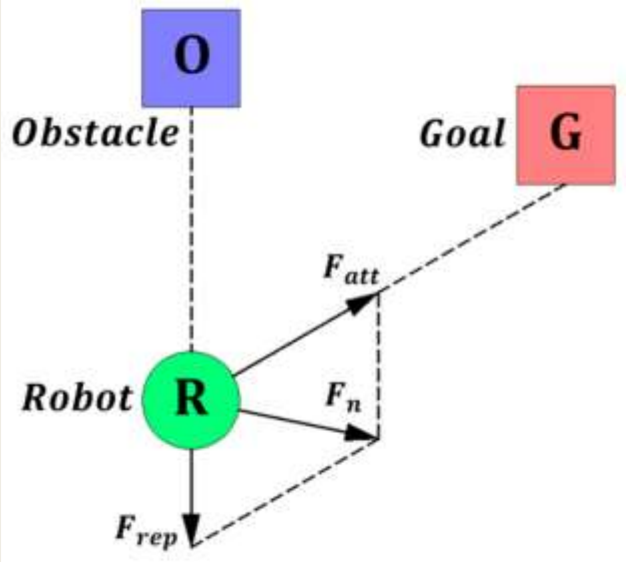
$$\mathbf{p}_{\text{best}} = \underset{\mathbf{p} \in \text{safe\_candidates}}{\arg\max} (d(p_i))$$

- p_{best}: optimal point

- p_i: potential points position
- Obs: set of all obstacles position
- d: Euclidean distance

Artificial Potential Fields (APF)

The Artificial Potential Field (APF) is a widely used approach for modeling robot motion. In this approach, the robot is modelled as a point under the influence of virtual forces:



Artificial Potential Field

1 Attractive Force

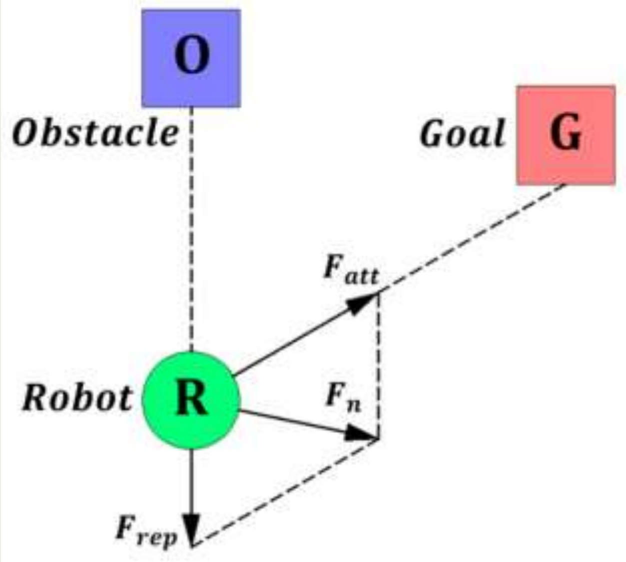
Aims to pull the robot towards a specified target location

2 Repulsive Force

Aims to repel the robot from both static obstacles (e.g., walls) and dynamic entities (e.g., pedestrians) that come too close

Artificial Potential Fields (APF)

The Artificial Potential Field (APF) is a widely used approach for modeling robot motion. In this approach, the robot is modelled as a point under the influence of virtual forces:



Artificial Potential Field

1 Attractive Force

Aims to pull the robot towards a specified target location

$$
\begin{aligned}
F_{att}(q) &= -\nabla \\
&= -\frac{1}{2} k_{att} \rho^2(q) \\
&= -k_{att}(q - q_d)
\end{aligned}
$$

- k_{att} = positive constant controlling the magnitude of the pull.
- q = current position vector of the robot.
- q_d = current position vector of the target.
- $\rho_{goal}(q) = \|q - q_d\|$ is the Euclidean distance from the robot's position to the goal position.
- $F_a(q)$ is a direct vector toward q_d with magnitude linearly related to the distance from q to q_d.

Artificial Potential Fields (APF)

The Artificial Potential Field (APF) is a widely used approach for modeling robot motion. In this approach, the robot is modelled as a point under the influence of virtual forces:

2 Repulsive Force

Aims to repel the robot from both static obstacles (e.g., walls) and dynamic entities (e.g., pedestrians) that come too close

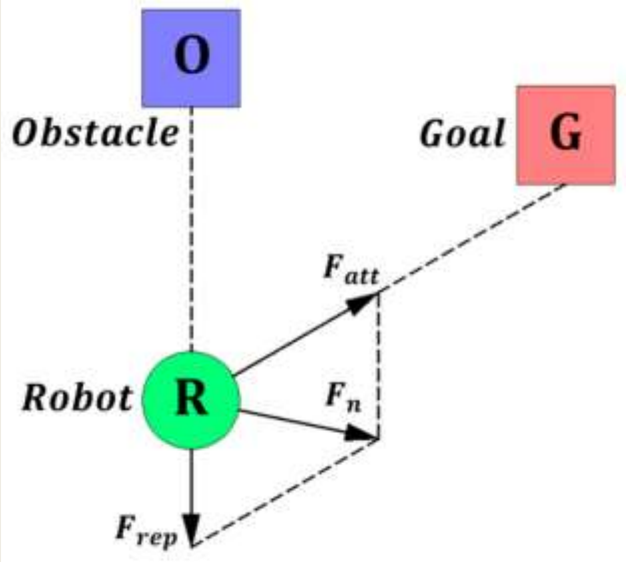
$$F_{\text{rep}}(q) = \begin{cases} k_{rep}\left(\frac{1}{d(q)} - \frac{1}{d_0}\right)\left(\frac{1}{d^2(q)}\right)\frac{q-q_{obs}}{\|q-q_{obs}\|}, & d(q) \leq d_0 \\ 0, & d(q) \geq d_0 \end{cases}$$

$$\Downarrow$$

$$F_{rep}^{total} = [\sum_{i=1}^{5} F_{\text{rep}}^{i}(q)] + F_{\text{rep}}^{ped}(q)$$

We take the 5 (at most) closest static obstacles

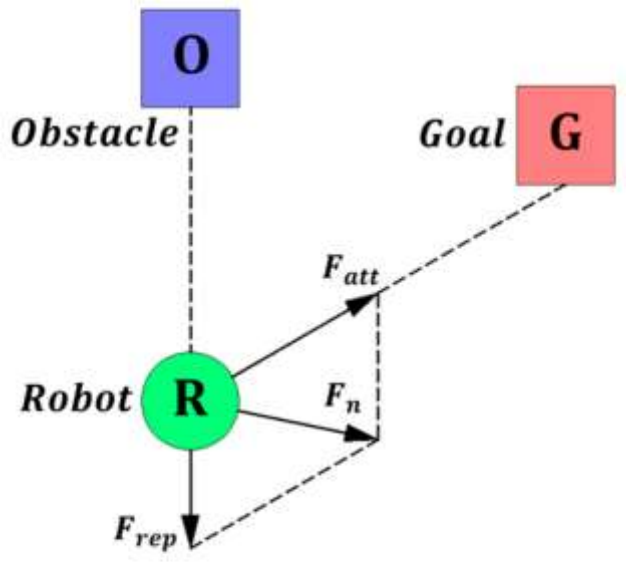
- k_{rep} = repulsive gain.
- $d(q)$ = distance from the robot to the obstacle.
- d_0 = distance of the obstacle repulsive force field.
- $d = \|q - q_{obs}\|$ as the distance between the robot and obstacles (static or dynamic)
- $F_{\text{rep}}^{i}(q)$ is the repulsive force from a static obstacle.
- $F_{\text{rep}}^{ped}(q)$ is the repulsive force from a pedestrian.



Artificial Potential Field

Artificial Potential Fields (APF)

The Artificial Potential Field (APF) is a widely used approach for modeling robot motion. In this approach, the robot is modelled as a point under the influence of virtual forces:



Artificial Potential Field

3 Total Force

The Total Force is the vector sum of the attraction and repulsion of the robot

$$F_{\text{total}} = F_{\text{att}} + F_{rep}^{total}$$

4 Velocities

The Total Force determines the robot's movement direction and speed

- Desired Velocity: $V_{desired} = F_{total}$
- Speed: $u = \|V_{desired}\|$
- Desired heading angle: $\theta_{desired} = \arctan2(V_{desired_y}, V_{desired_x})$
- Heading error: $\theta_{error} = \theta_{desired} - \theta_{robot}$
- Robot orientation: θ_{robot}

➡

$$u_{linear} = \begin{cases} u\cos(\theta_{error}), & u_{linear} \le u_{linear}^{max} \\ u_{linear}^{max}, & u_{linear} \ge u_{linear}^{max} \end{cases}$$

➡

- $\theta_{error} = 0 => u_{linear} = u$
- $\theta_{error} = \pm\frac{\pi}{2} => u_{linear} = 0$
- $\theta_{error} > 90^\circ => u_{linear} < 0$

➡

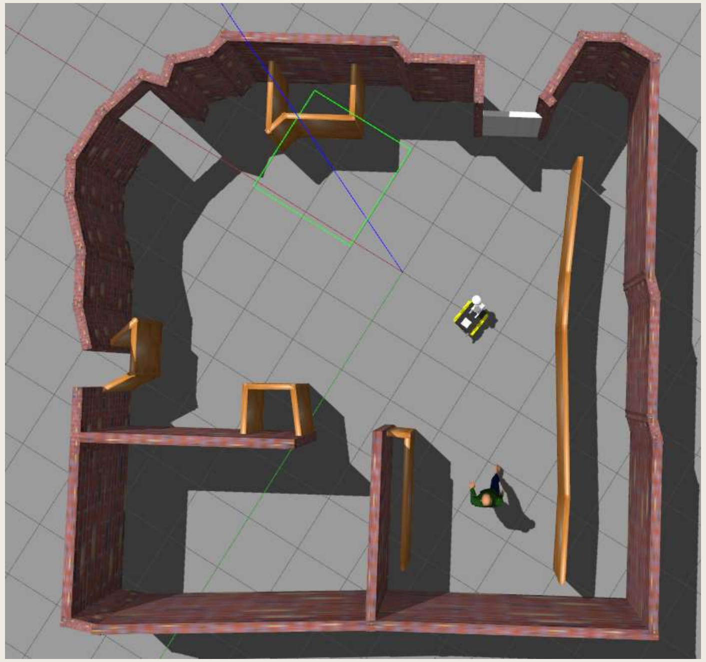
$$u_{angular} = \begin{cases} k_{att}\theta_{error}, & u_{angular} \le u_{angular}^{max} \\ u_{angular}^{max}, & u_{angular} \ge u_{angular}^{max} \end{cases}$$

➡

- $\theta_{error} > 0 =>$ rotate left
- $\theta_{error} < 0 =>$ rotate right
- $\theta_{error} = 0 =>$ don't rotate

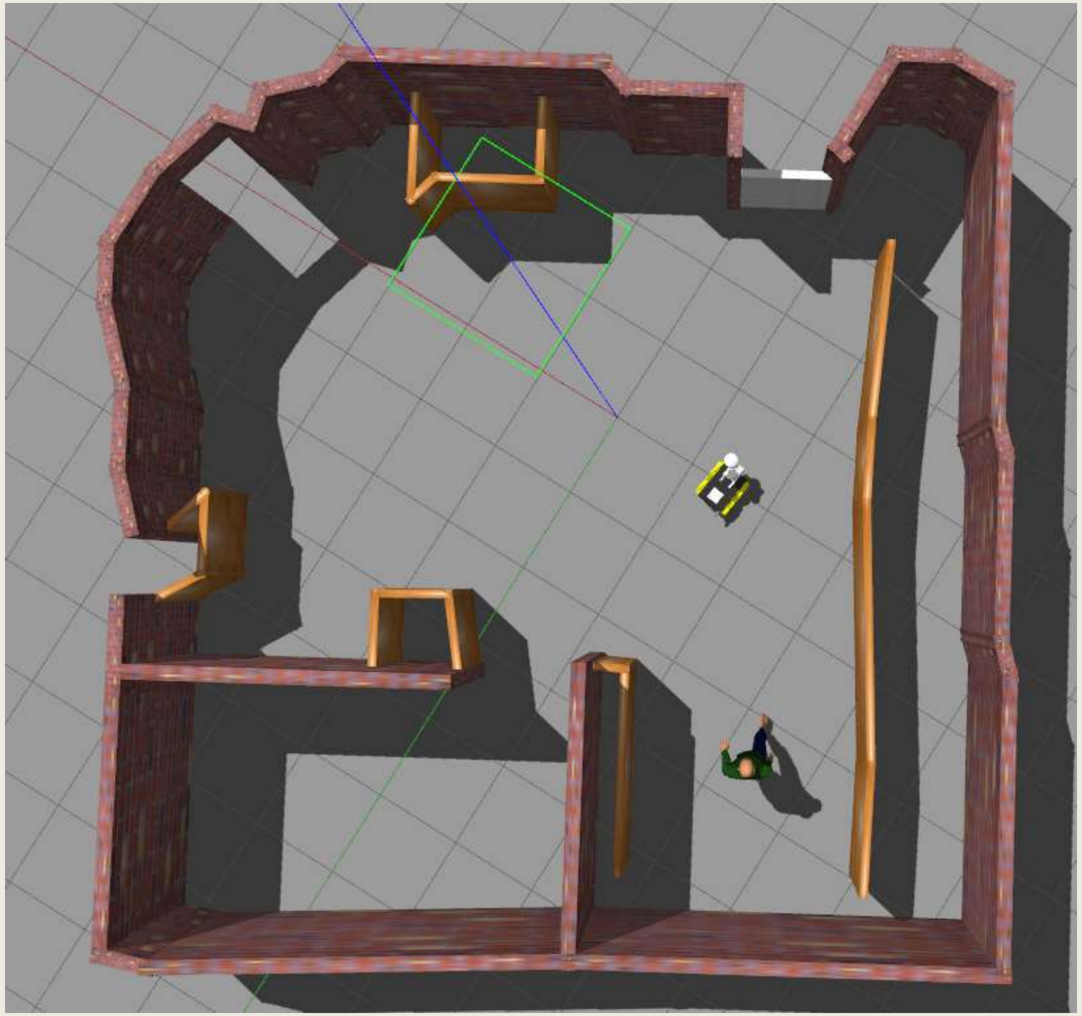
Simulations

CASE-A



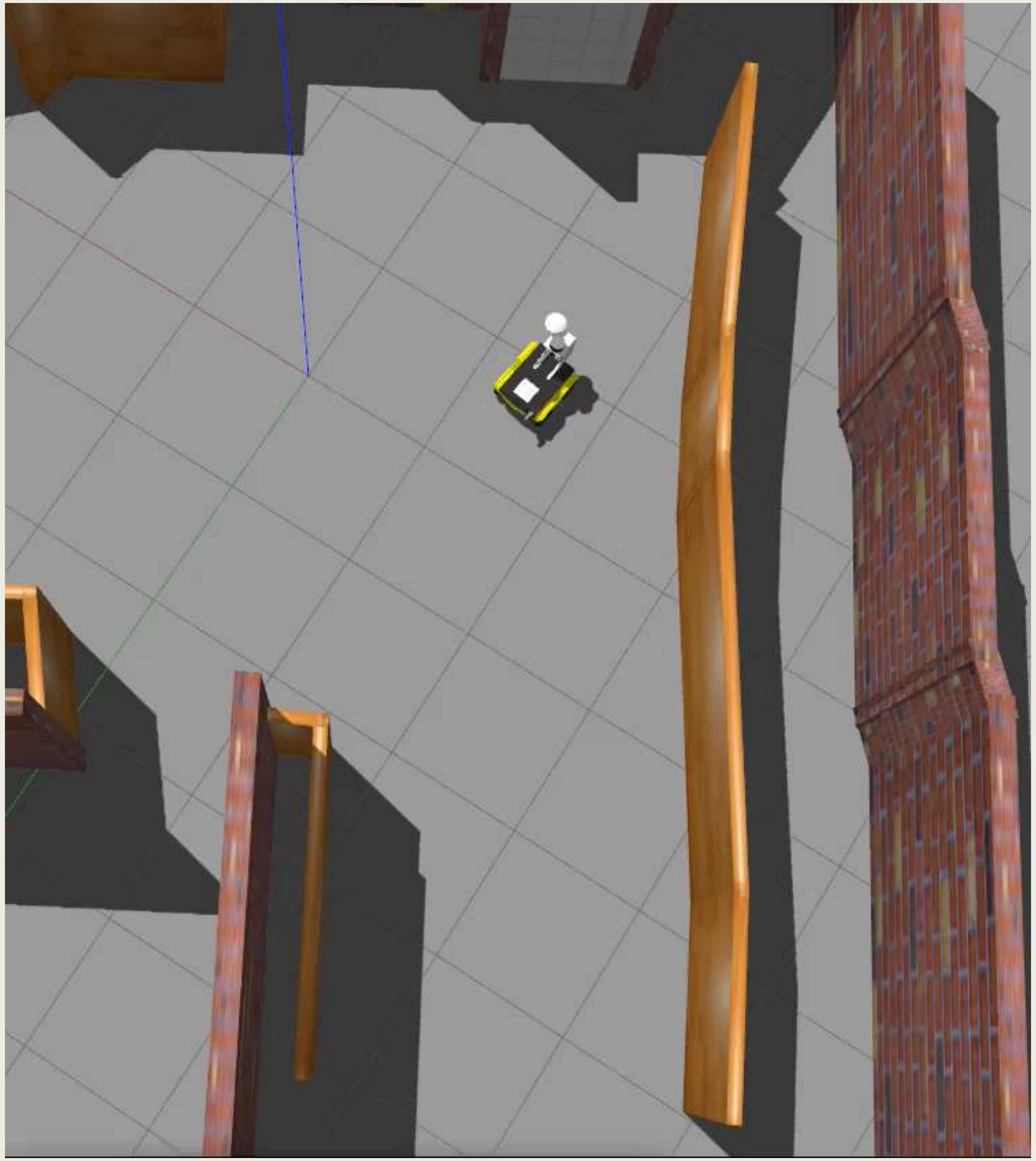
CASE-B

Simulations



Case A: The pedestrian is walking directly towards the robot, with the robot itself slightly angled relative to pedestrian's path.

Simulations



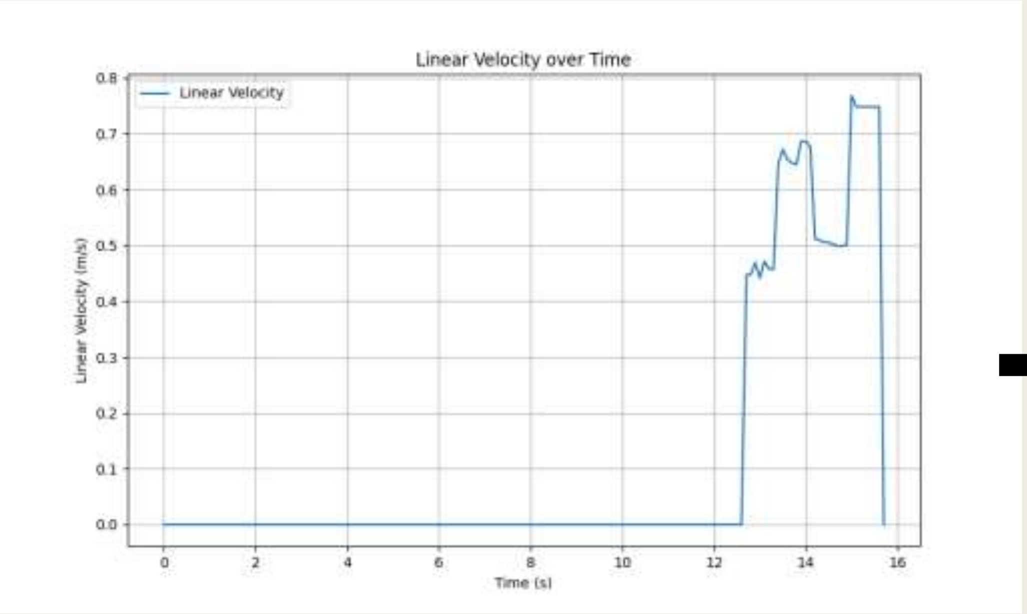
Case A (video)

Case A: The pedestrian is walking directly towards the robot, with the robot itself slightly angled relative to pedestrian’s path.

Simulations

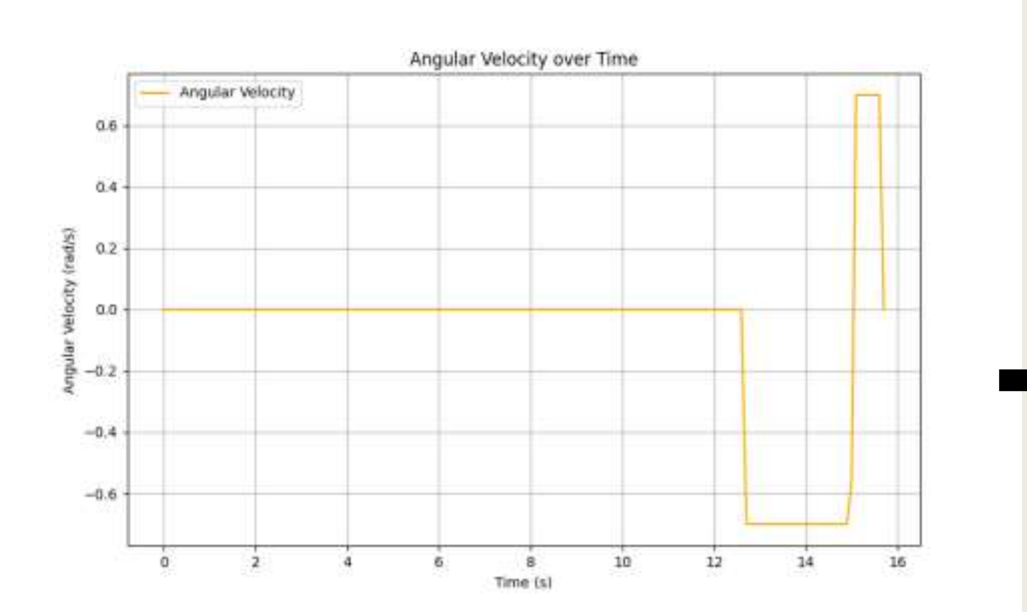
Case A: The pedestrian is walking directly towards the robot, with the robot itself slightly angled relative to pedestrian's path.

	Linear	Angular
> 0	forward	left
< 0	backward	right



Linear Velocity - Time

Linear Velocity - Time:
- **Blue line:** linear velocity

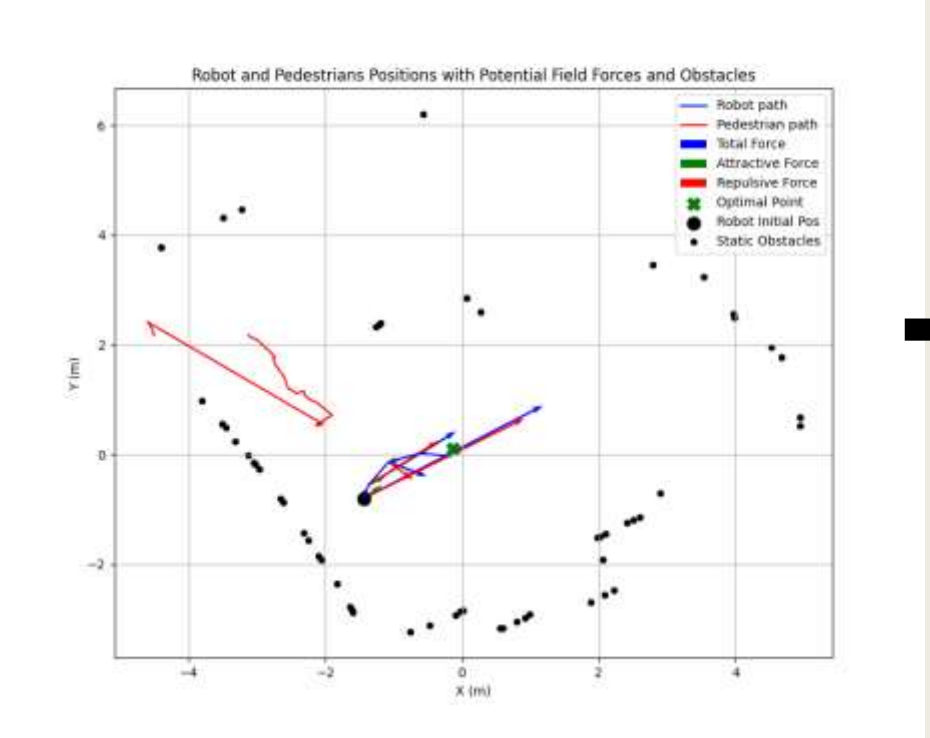


Angular Velocity - Time

Angular Velocity - Time:
- **Orange line:** angular velocity

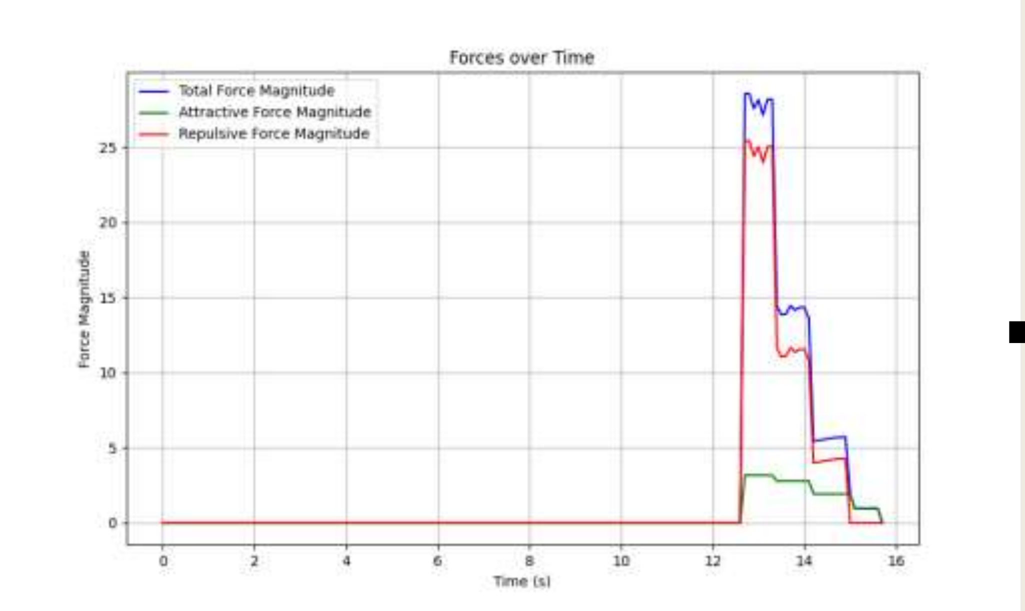
Simulations

Case A: The pedestrian is walking directly towards the robot, with the robot itself slightly angled relative to pedestrian's path.



Forces in the Map (APF):

- **Blue line**: robot path
- **Red line:** pedestrian path
- **Blue arrow:** total force
- **Green arrow:** attractive force
- **Red arrow:** repulsive force
- **Green cross:** optimal point
- **big black circle:** robot initial position
- **Small black circles**: static obstacles

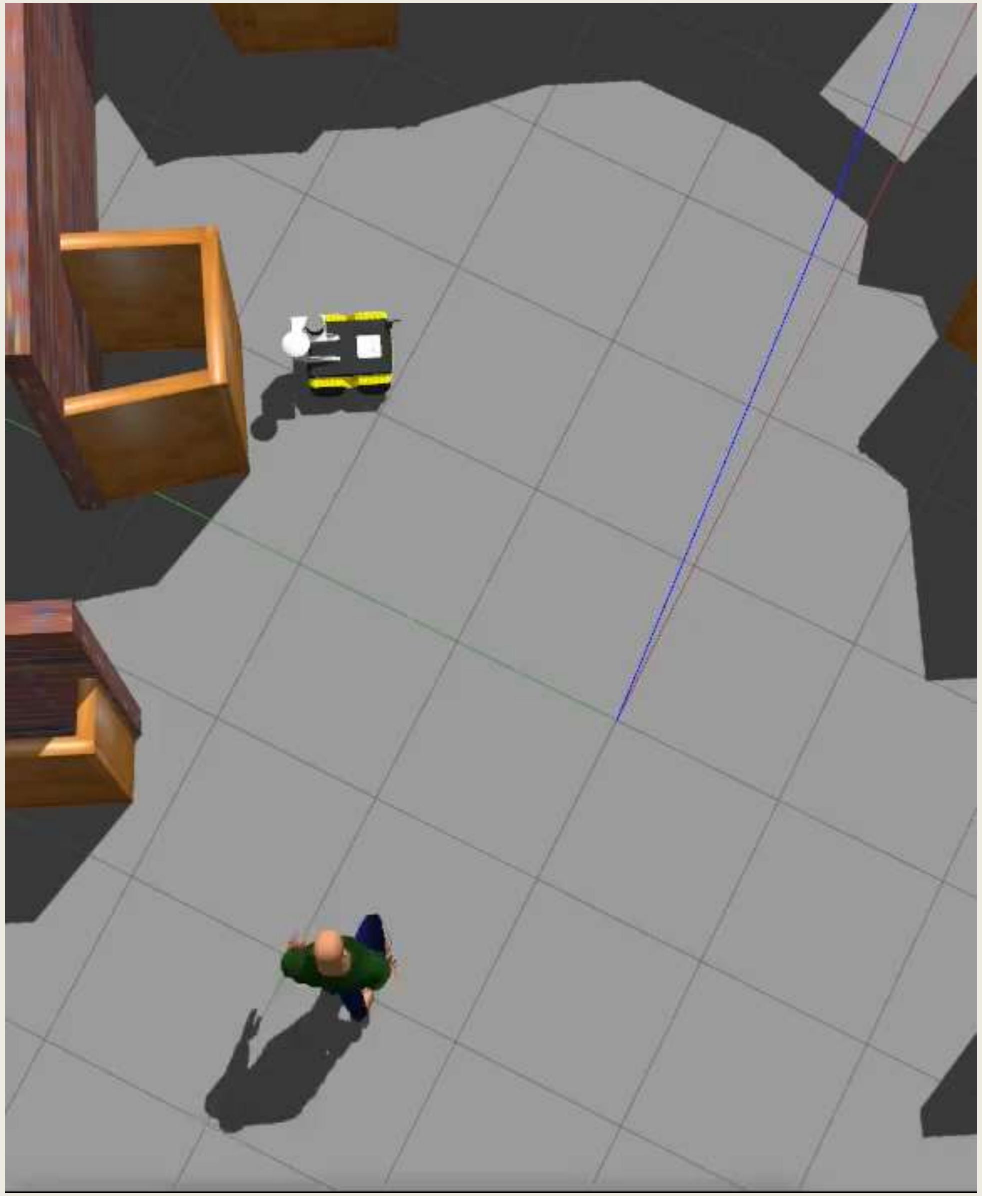


Forces over Time (APF) :

- **Blue line:** total force
- **Red line:** repulsive force
- **Green line:** attractive force

Simulations

Case B: The pedestrian is walking directly towards the robot, and the robot is actively rotating. Unlike the previous simulation, now we have the wall behind the robot.



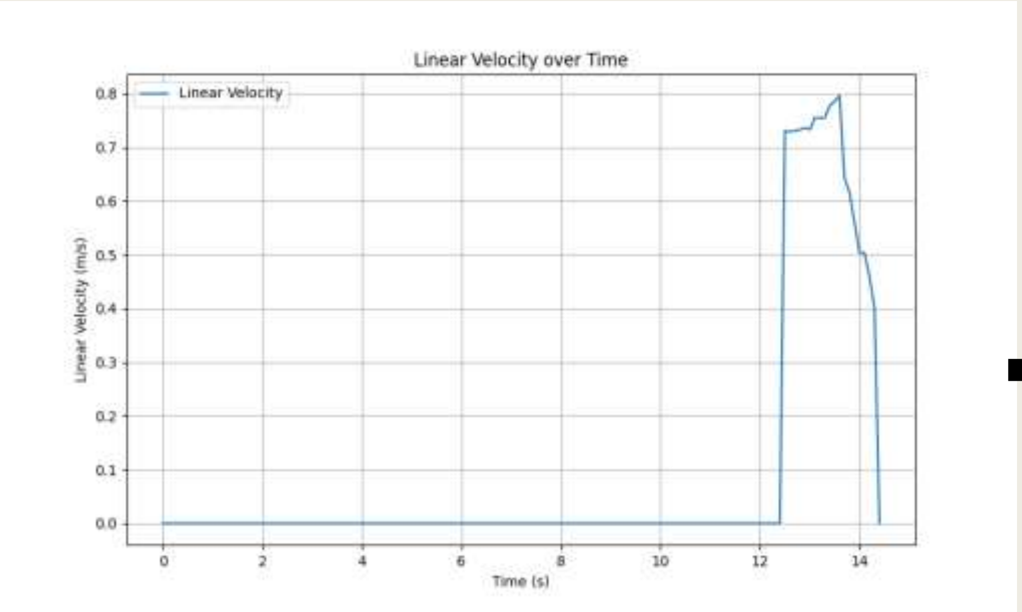
Case B (video)

Simulations

Case B: The pedestrian is walking directly towards the robot, and the robot is actively rotating. Unlike the previous simulation, now we have the wall behind the robot.

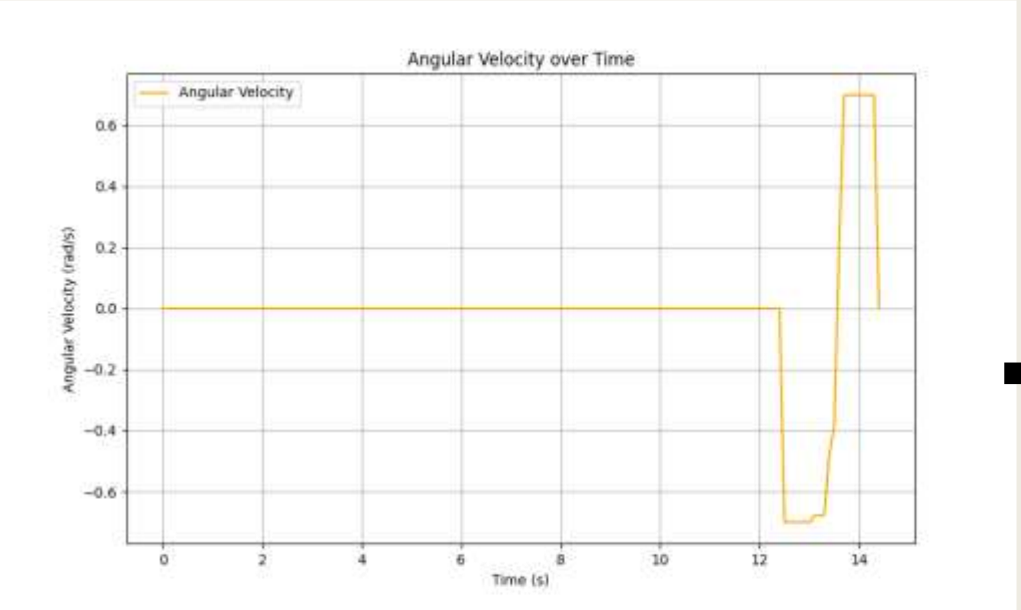
Simulations

Case B: The pedestrian is walking directly towards the robot, and the robot is actively rotating. Unlike the previous simulation, now we have the wall behind the robot.



Linear Velocity - Time

Linear Velocity - Time:
- **Blue line:** linear velocity



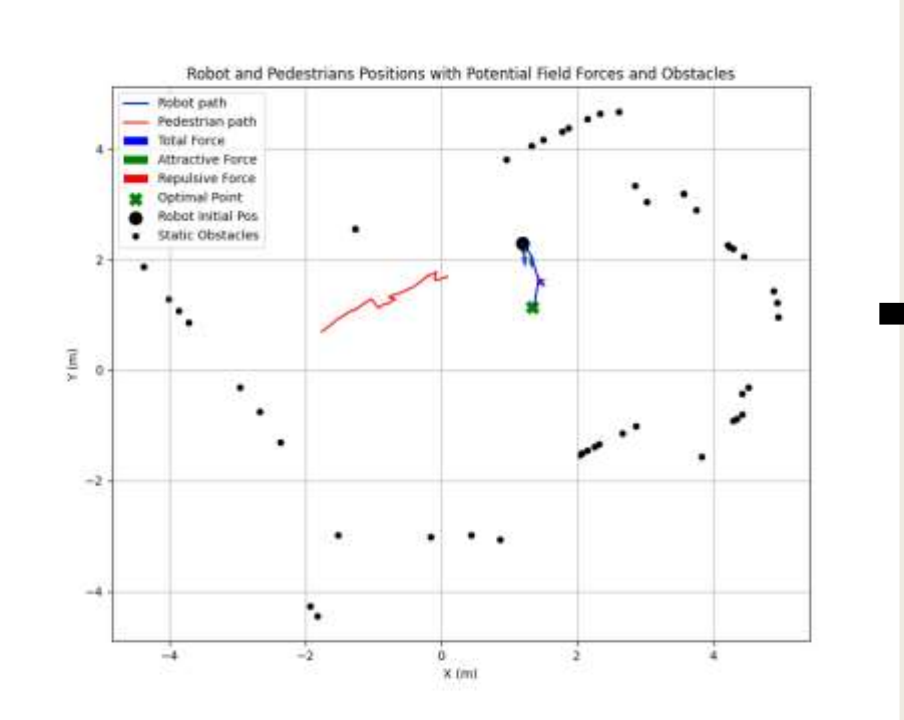
Angular Velocity - Time

Angular Velocity - Time:
- **Orange line:** angular velocity

	Linear	Angular
> 0	forward	left
< 0	backward	right

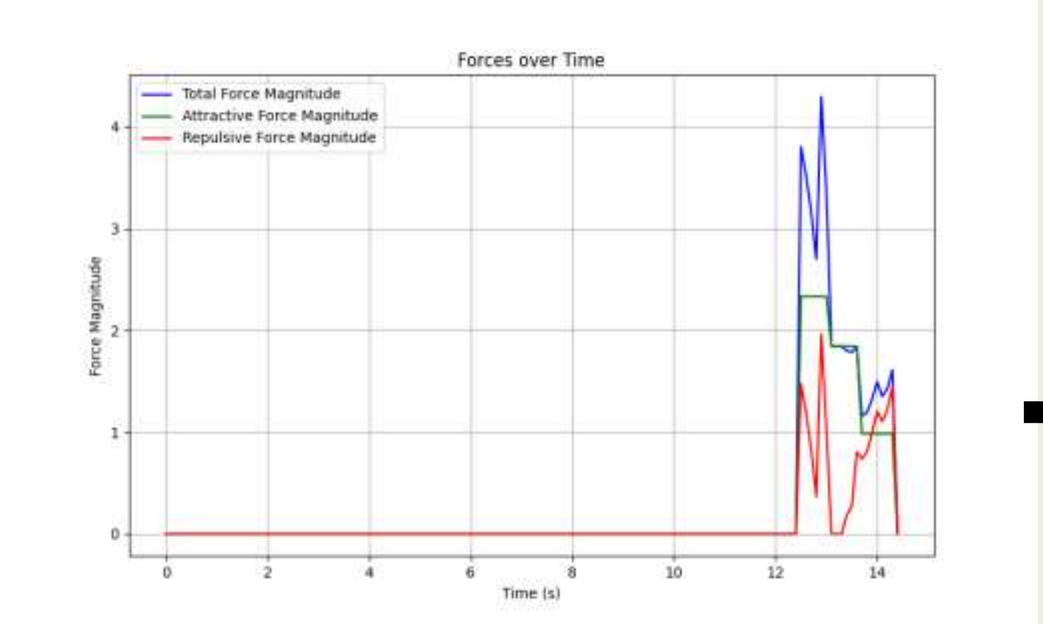
Simulations

Case B: The pedestrian is walking directly towards the robot, and the robot is actively rotating. Unlike the previous simulation, now we have the wall behind the robot.



Forces in the Map (APF):

- **Blue line:** robot path
- **Red line:** pedestrian path
- **Blue arrow:** total force
- **Green arrow**: attractive force
- **Red arrow:** repulsive force
- **Green cross:** optimal point
- **big black circle**: robot initial position
- **Small black circles:** static obstacles



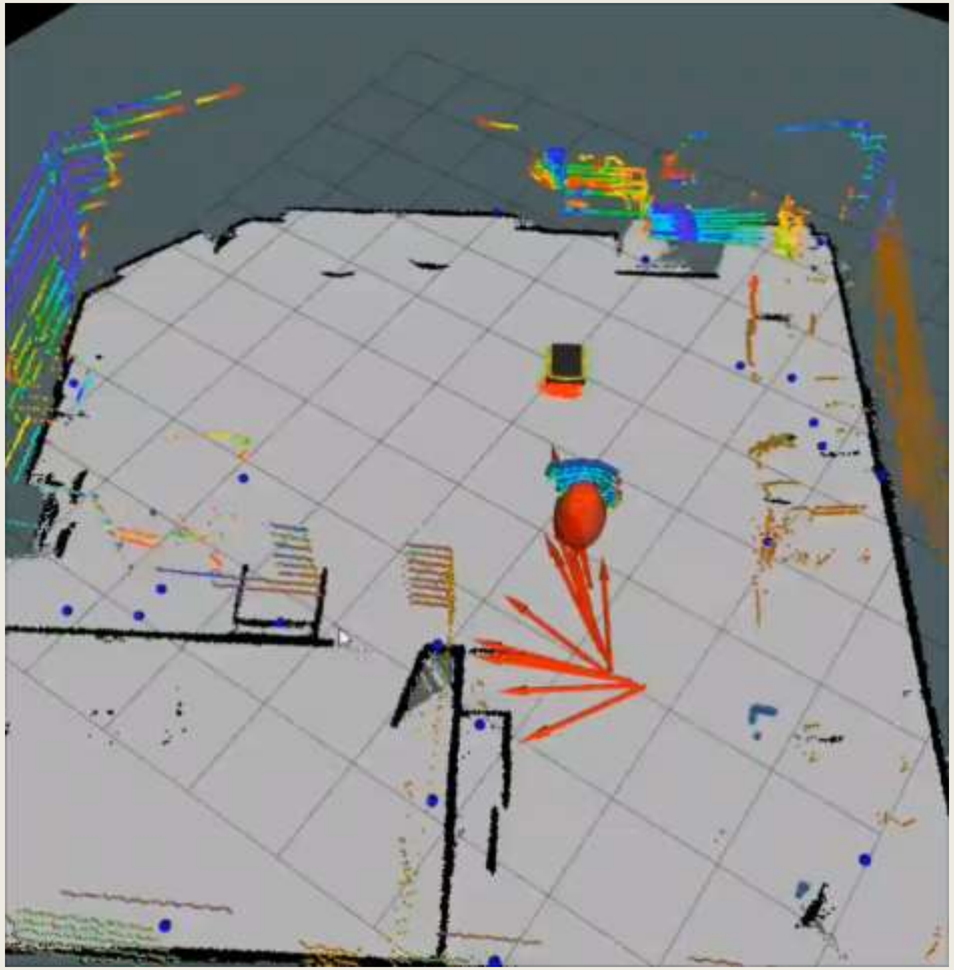
Forces over Time (APF):

- **Blue line: t**otal force
- **Red line: r**epulsive force
- **Green line:** attractive force

Experimental Results – Lab Results

Lab experiment (video)

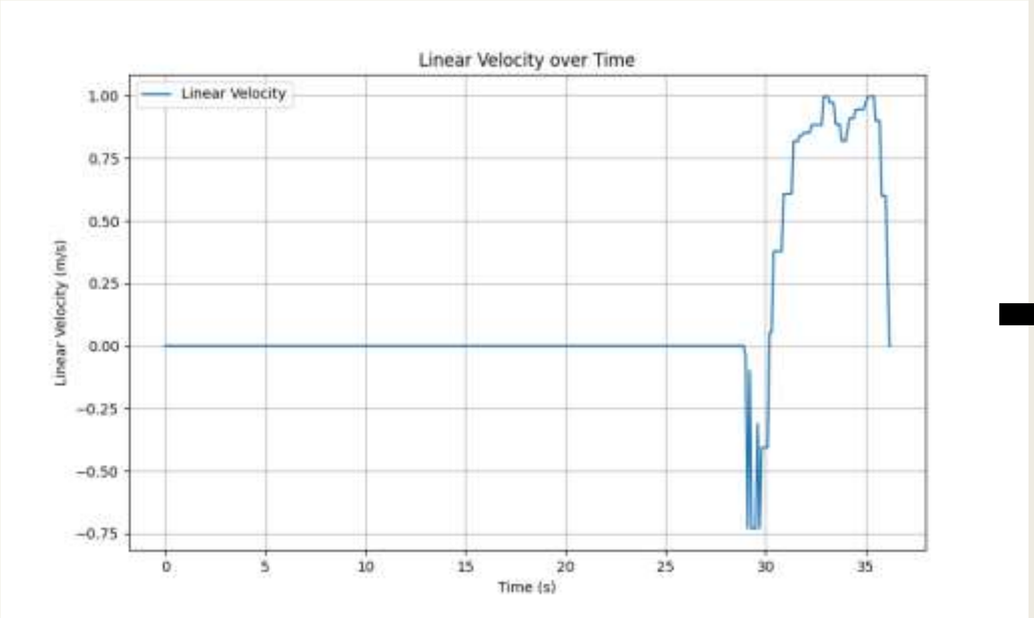
=



Lab experiment (RVIZ)

Experimental Results

Case A: The pedestrian is walking directly towards the robot, and the robot orientation is aligned with the pedestrian’s approach.



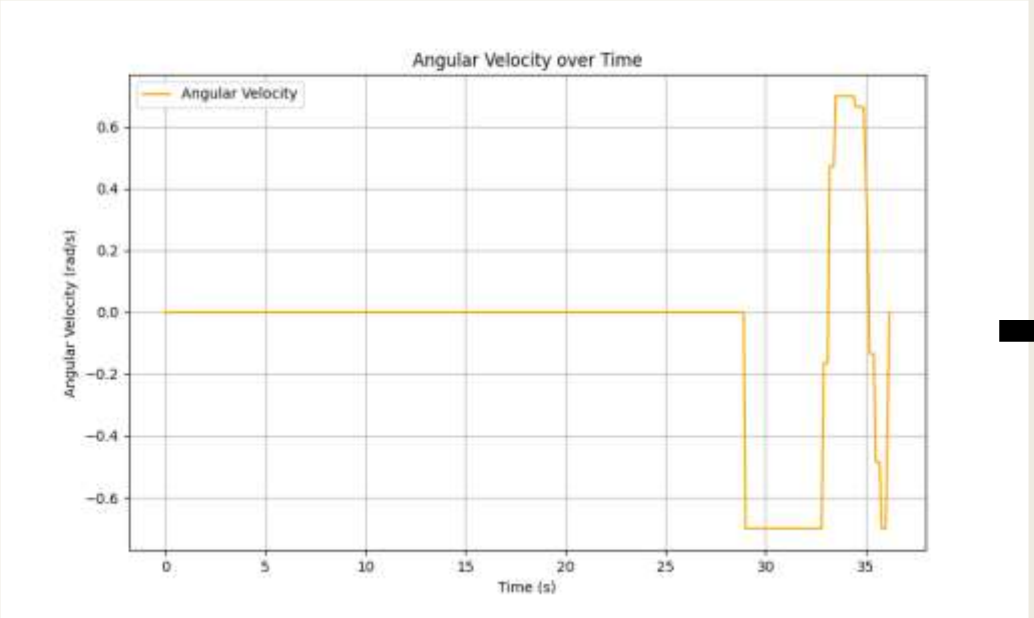
Linear Velocity - Time

Linear Velocity - Time:
- **Blue line:** linear velocity

	Linear	Angular
> 0	forward	left
< 0	backward	right

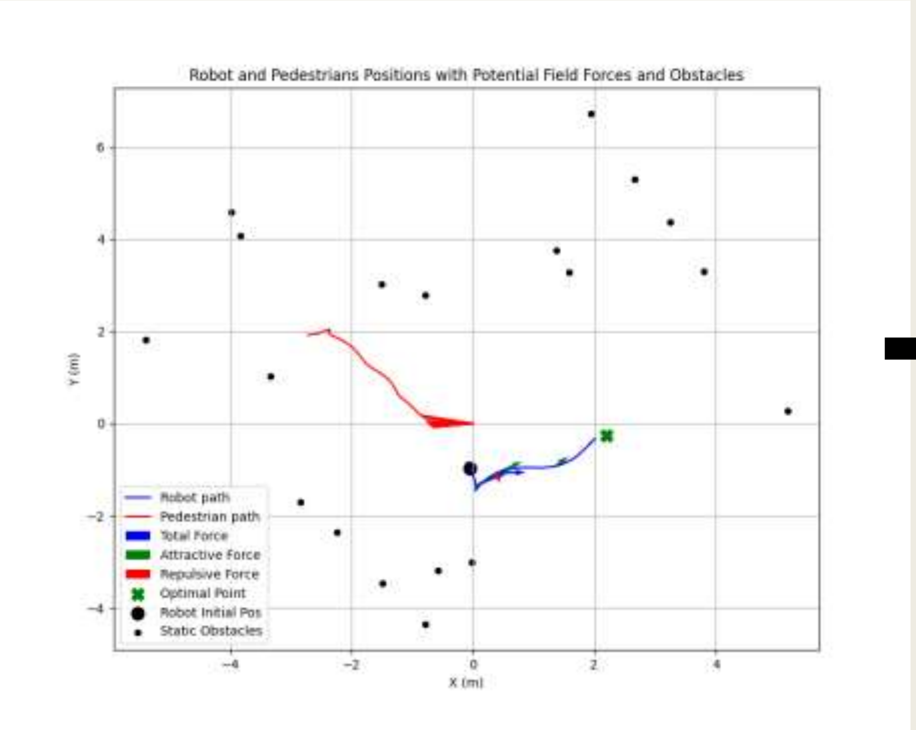
Experimental Results

Case A: The pedestrian is walking directly towards the robot, and the robot orientation is aligned with the pedestrian's approach.



Angular Velocity - Time

Angular Velocity - Time:
- **Orange line:** angular velocity

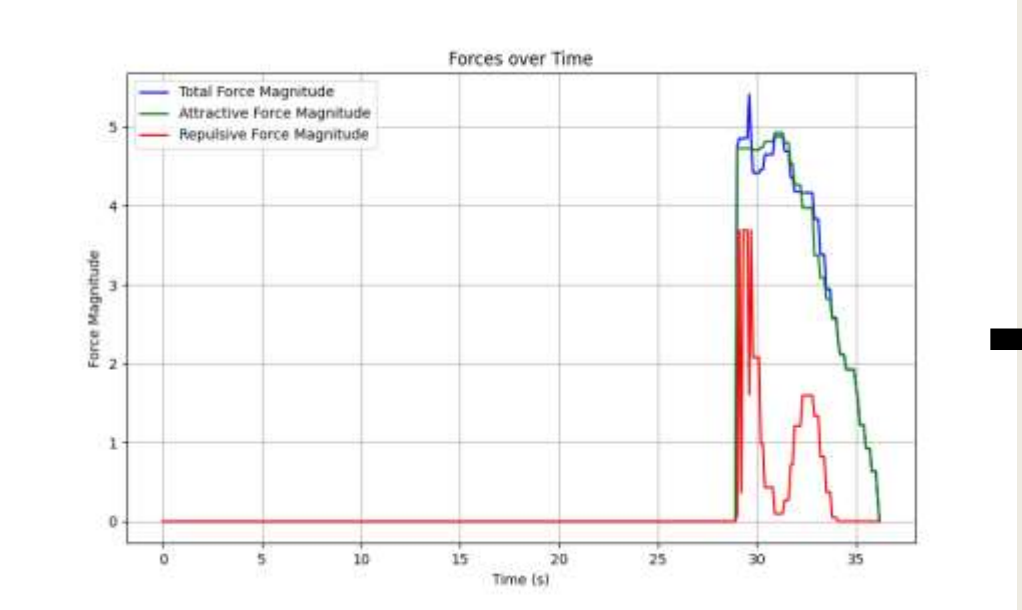


Forces in the Map (APF):

- **Blue line:** robot path
- **Red line:** pedestrian path
- **Blue arrow:** total force
- **Green arrow:** attractive force
- **Red arrow:** repulsive force
- **Green cross:** optimal point
- **big black circle:** robot initial position
- **Small black circles:** static obstacles

Experimental Results

Case A: The pedestrian is walking directly towards the robot, and the robot orientation is aligned with the pedestrian's approach.

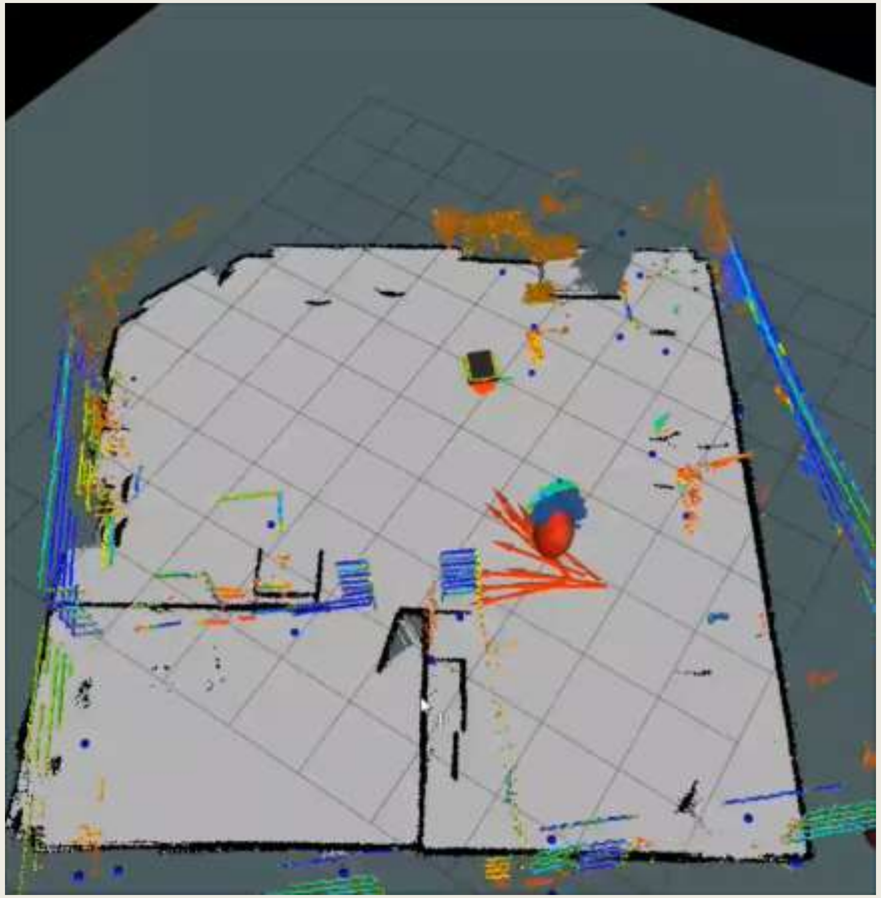


Forces over Time (APF):

- **Blue line:** total force
- **Red line:** repulsive force
- **Green line:** attractive force

Lab experiment (video)

=



Lab experiment (RVIZ)

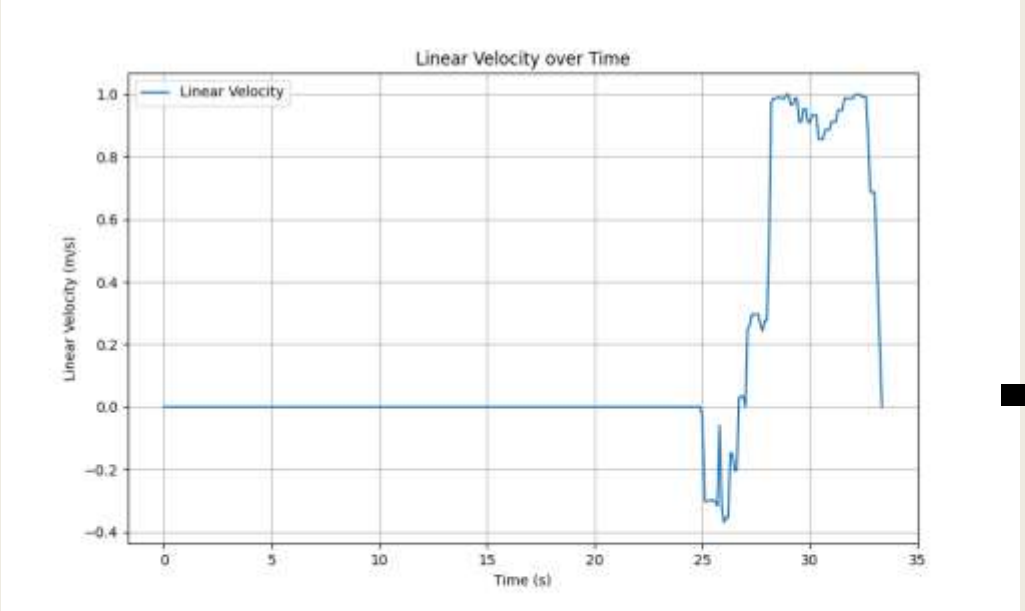
Experimental Results

Case B: The pedestrian is walking directly towards the robot, and the robot is actively rotating. Unlike the previous simulation, now we have desk at left of the robot.

Experimental Results

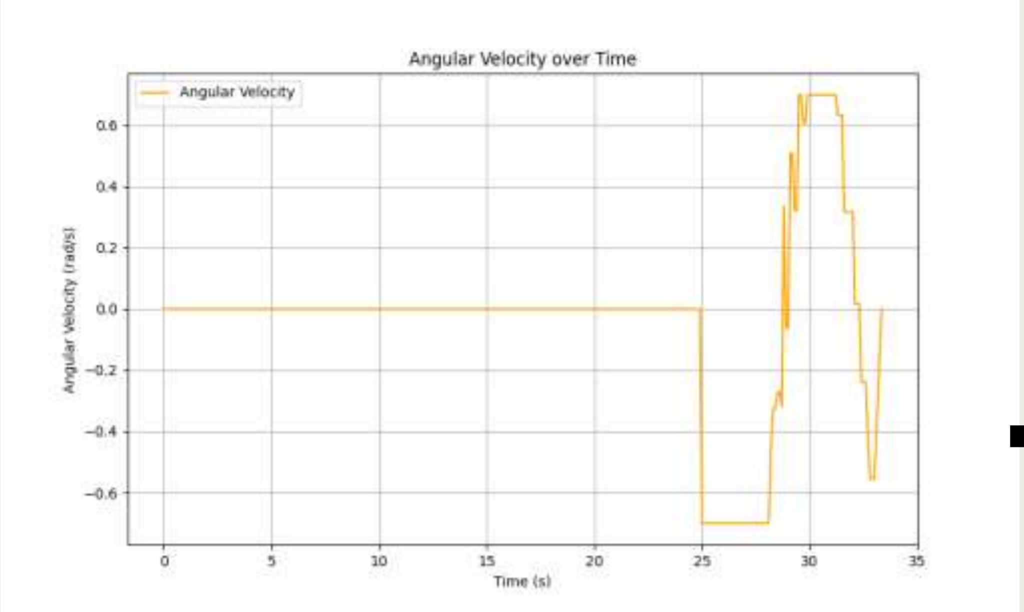
Case B: The pedestrian is walking directly towards the robot, and the robot is actively rotating. Unlike the previous simulation, now we have desk at left of the robot.

	Linear	Angular
> 0	forward	left
< 0	backward	right



Linear Velocity - Time

Linear Velocity - Time:
- **Blue line:** linear velocity

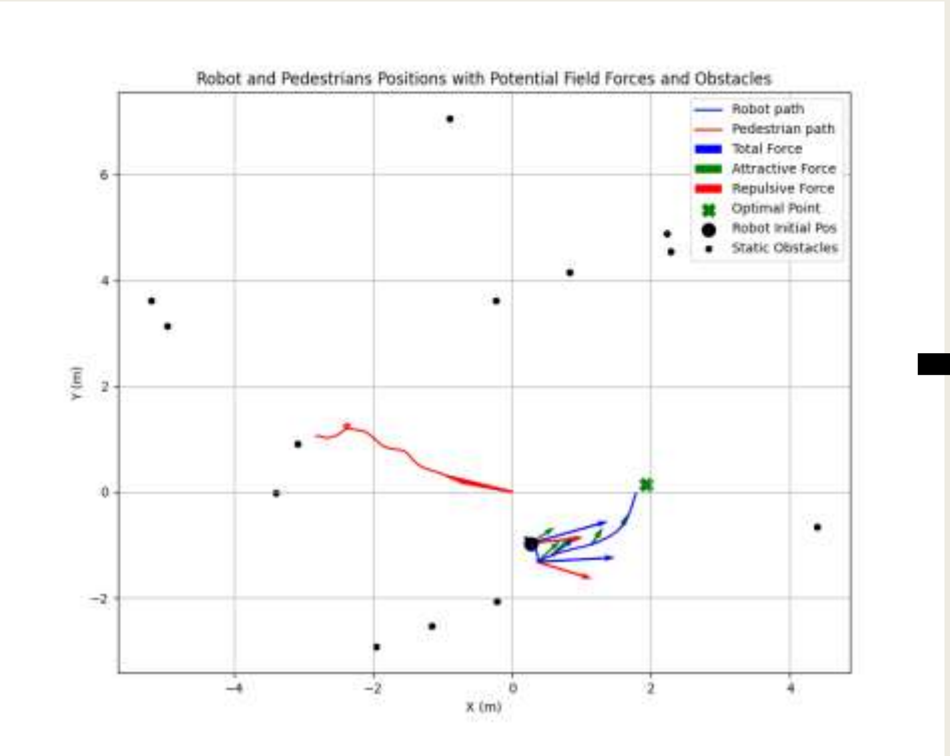


Angular Velocity - Time

Angular Velocity - Time:
- **Orange line:** angular velocity

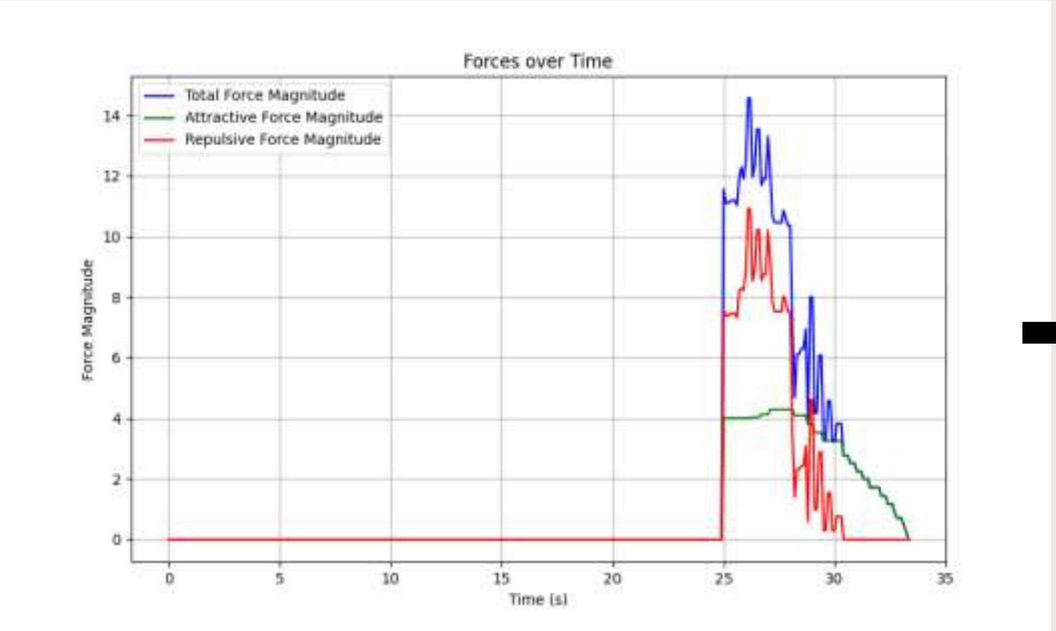
Experimental Results

Case B: The pedestrian is walking directly towards the robot, and the robot is actively rotating. Unlike the previous simulation, now we have desk at left of the robot.



Forces in the Map (APF):

- **Blue line:** robot path
- **Red line:** pedestrian path
- **Blue arrow:** total force
- **Green arrow:** attractive force
- **Red arrow:** repulsive force
- **Green cross:** optimal point
- **big black circle:** robot initial position
- **Small black circles**: static obstacles



Forces over Time (APF):

- **Blue line:** total force
- **Red line:** repulsive force
- **Green line:** attractive force

Strengths & Weakness

Strengths

- Real-time pedestrian detection and avoidance (under-constraint)
- Works well in dynamic environments
- Modular ROS design, easily extendable

Weakness

- Struggles with wi-fi connection in lab experiments, so that it delays to send the LiDAR data.
- Local minima issue in potential fields

Potential Improvements

- Extension of the code to detect multiple pedestrians at the same time
- Better detection algorithm of dynamic and static data points from LiDAR
- Solution for local minimum problem in Potential Field approach
- Integration of additional sensing modalities, such as 3D cameras

Thank you for your time and interest in our work.