# Marker-based FastSLAM on the AlphaBot2

## Autonomous Systems Project 2023/2024


J. Pessoa
ist198915

J. Gonçalves
ist199995

M. Ribeiro
ist196446

T. Nogueira
ist1100029

Group 27

Instituto Superior Técnico

# Presentation Overview

# Simulation Models and Algorithms

In our micro-simulator, we define the **motion**, **measurement**, and **sensor models**, along with an initial state $X(0)$ and control sequence $U(t)$, to generate synthetic data.

# Motion Model

Given the absence of wheel odometry, we make use of the **velocity motion model**:

$$\begin{pmatrix} x' \\ y' \\ \theta' \end{pmatrix} = \begin{pmatrix} x \\ y \\ \theta \end{pmatrix} + \begin{pmatrix} -\frac{\hat{v}}{\hat{\omega}} \sin\theta + \frac{\hat{v}}{\hat{\omega}} \sin(\theta + \hat{\omega}\Delta t) \\ \frac{\hat{v}}{\hat{\omega}} \cos\theta - \frac{\hat{v}}{\hat{\omega}} \cos(\theta + \hat{\omega}\Delta t) \\ \hat{\omega}\Delta t + \hat{\gamma}\Delta t \end{pmatrix}$$

Where $\hat{v}$ and $\hat{\omega}$ model **real motion!** (noisy values).

```
% Add some Gaussian random noise to the movement.
v = normrnd(movement(1), variance(1));
w = normrnd(movement(2), variance(2));

% Check if there are no singularities
% ...

% Calculate the new position
delta = zeros(3,1);
delta(1) = -(v/w) * sin(pos(3)) + (v/w)* sin(pos(3) + w * dt);
delta(2) = (v/w) * cos(pos(3)) - (v/w)* cos(pos(3) + w * dt);
delta(3) = w * dt;

newpos = pos + delta;
```

**Note:** For singularities we can use L'Hôpital's rule

$$\lim_{x \to a^+} = \frac{f(x)}{g(x)} = \frac{f'(x)}{g'(x)}$$

# Measurement Model

# Simulation Models
## Measurement Model

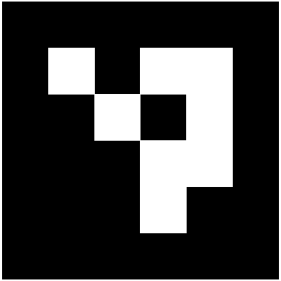

Figure: ArUco marker

**Translation:**

- `x`: Fiducial x-coordinate relative to the camera.
- `y`: Fiducial y-coordinate relative to the camera.
- `z`: Fiducial z-coordinate relative to the camera.

**Header:**

- `frame_id`: Camera frame ID.
- `stamp`: Message timestamp.

**Child Frame ID:**

- `fiducial_<id>`: Unique identifier.

Sensors usually provide the marker’s relative Cartesian coordinates or a range and bearing. These (noisy) values can be transformed into the **world reference frame**:

```
% Transform to world coordinates
particles(pIdx).landmarks(lIdx).pos = [
    particles(pIdx).position(1) + x;
    particles(pIdx).position(2) + y;
];
```

```
% Transform to world coordinates
particles(pIdx).landmarks(Id).pos = [
    particles(pIdx).position(1) + cos(angle) * distance;
    particles(pIdx).position(2) + sin(angle) * distance;
];
```

Assuming that the robot's pose is $s_t = \langle s_{t,x}, s_{t,y}, s_{t,\theta} \rangle$ and the marker's position is written as $l_{n_t} = \langle l_{n_t,x}, l_{n_t,y} \rangle$.

(1) Cartesian coordinates:

$$g = \begin{bmatrix} l_{n_t,x} \\ l_{n_t,y} \end{bmatrix}$$

$$G = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

(2) Polar coordinates:

$$g = \begin{bmatrix} r(s_t, l_{n_t}) \\ \phi(s_t, l_{n_t}) \end{bmatrix}$$

$$G = \begin{bmatrix} \frac{l_{n_t,x} - s_{t,x}}{\sqrt{q}} & \frac{l_{n_t,y} - s_{t,y}}{\sqrt{q}} \\ \frac{s_{t,x} - l_{n_t,x}}{q} & \frac{l_{n_t,y} - s_{t,y}}{q} \end{bmatrix},$$

$$q = (l_{n_t,x} - s_{t,x})^2 + (l_{n_t,y} - s_{t,y})^2$$

# Sensor Model

- Based on official documentation, the Raspberry Pi Camera Module 2 has a field of view (FOV) as follows:

$$\text{FOV} = 2 \cdot \arctan\left(\frac{\text{sensor size}}{2 \cdot \text{focal length}}\right)$$

  - **Horizontal FOV**: Approximately 62.2 degrees
  - **Vertical FOV**: Approximately 48.8 degrees
- 14 cm ArUcos are roughly perceived up to 3 m, with the standard $1280 \times 720$ resolution (width $\times$ height).

# Resampling Algorithm

To prevent sample **impoverishment and degeneration** we resample only when the particle variance is low.

$$\mathrm{ESS} = \left( \sum_{i=1}^{N} (W_n^i)^2 \right)^{-1}$$

- Resampling is performed if the Effective Sample Size (ESS) is below the threshold "number of particles/2".

# Simulation on a 2D Environment

# Simulation on a 2D Environment

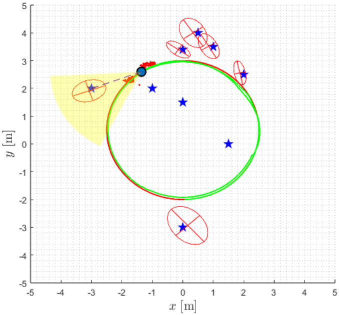


Figure: Noise variance is equal to the velocities

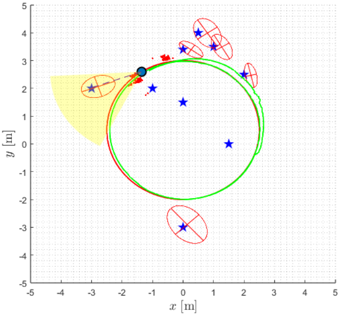


Figure: Noise variance is equal to 1.5× of the velocities

TÉCNICO LISBOA

# Simulation on a 3D Environment

# Simulation on a 3D Environment

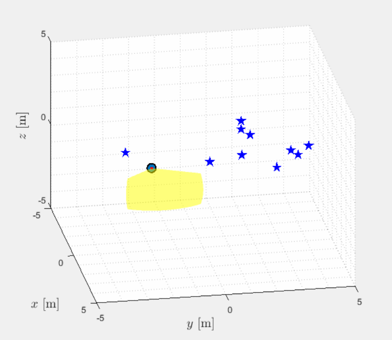


Figure: Noise variance is equal to the velocities (animation, gif)

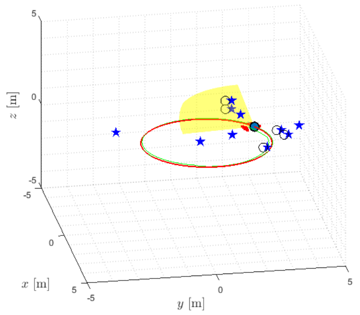


Figure: Noise variance is equal to $1.5\times$ of the velocities

# Simulation with Unknown Data Association

- What happens if we **disregard** (or don't have) the IDs provided by the `detect_aruco` function?

# Unknown Data Association
## Problem

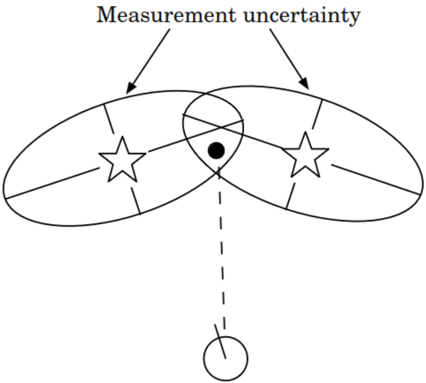


Figure: Measurement Ambiguity.

1. **Initialization**:
   - Determine the number of measurements and existing landmarks observed and initialize a cost table with zeros.
2. **Compute Cost Table**:

$$\text{cost} = \frac{1}{\sqrt{|2\pi Z_{n,t}|}} \exp\left\{-\frac{1}{2}(z_t - \hat{z}_{n,t})^T [Z_{n,t}]^{-1} (z_t - \hat{z}_{n,t})\right\}$$

3. **Assign Default Importance for New Landmarks**:
   - Fill remaining columns of the cost table with default importance for new landmarks.
4. **Data Association**:
   - For each measurement, determine the best matching landmark and update particle's weight.
5. **Weight Update**:
   - Update particle's weight based on the combined likelihood of all measurements.

| Measurement | Landmark 1 | Landmark 2 | New Landmark 1 | New Landmark 2 | New Landmark 3 |
|---|---|---|---|---|---|
| 1 | 0.8 | 0.3 | 0.1 | 0.1 | 0.1 |
| 2 | 0.5 | 0.7 | 0.1 | 0.1 | 0.1 |
| 3 | 0.01 | 0.01 | 0.1 | 0.1 | 0.1 |

**Data Association**:

- Measurement 1: Best match is Landmark 1 (0.8)
- Measurement 2: Best match is Landmark 2 (0.7)
- Measurement 3: Best match is New Landmark 1 (0.1)

# Unknown Data Association
## Example of a Simulated Result

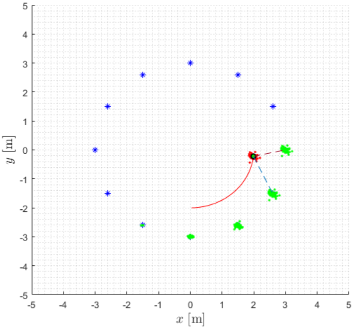


Figure: 2D simulation with unknown data association

# Gazebo Simulation

# Gazebo Simulation
## AlphaBot2 Xacro Model in Gazebo

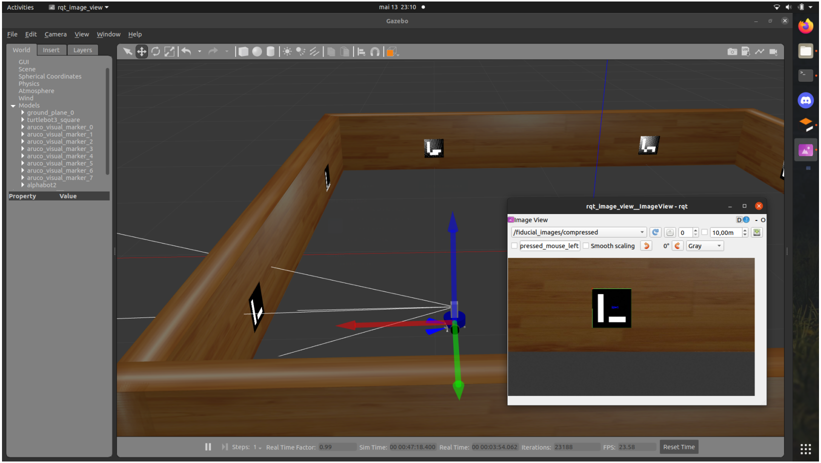


Figure: ROS node working with the simulated robot in Gazebo

# Gazebo Simulation

## Simulated Detection of an ArUco

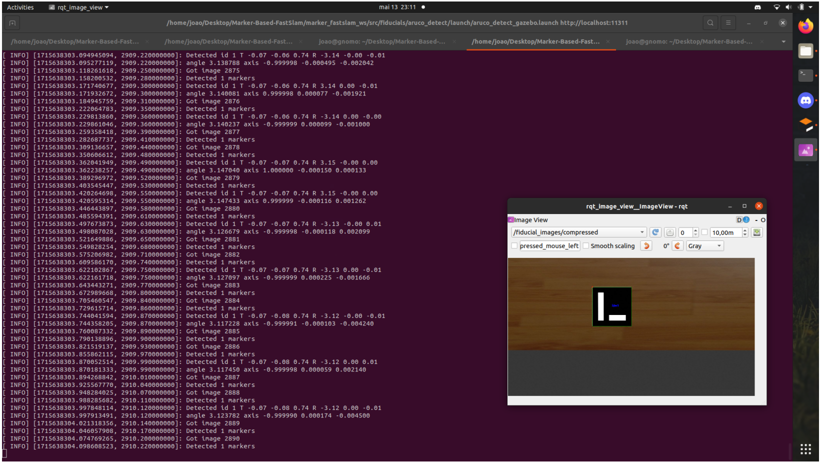


Figure: ROS node working with the simulated robot in Gazebo

# Questions

# Thank You!