Università di Bologna

Master Degree in Automation Engineering

TOPIC HIGHLIGHT PROJECT

Targets Detection, Classification and Localization with TurtleBot 3

Professor:
Silvia Ferrari

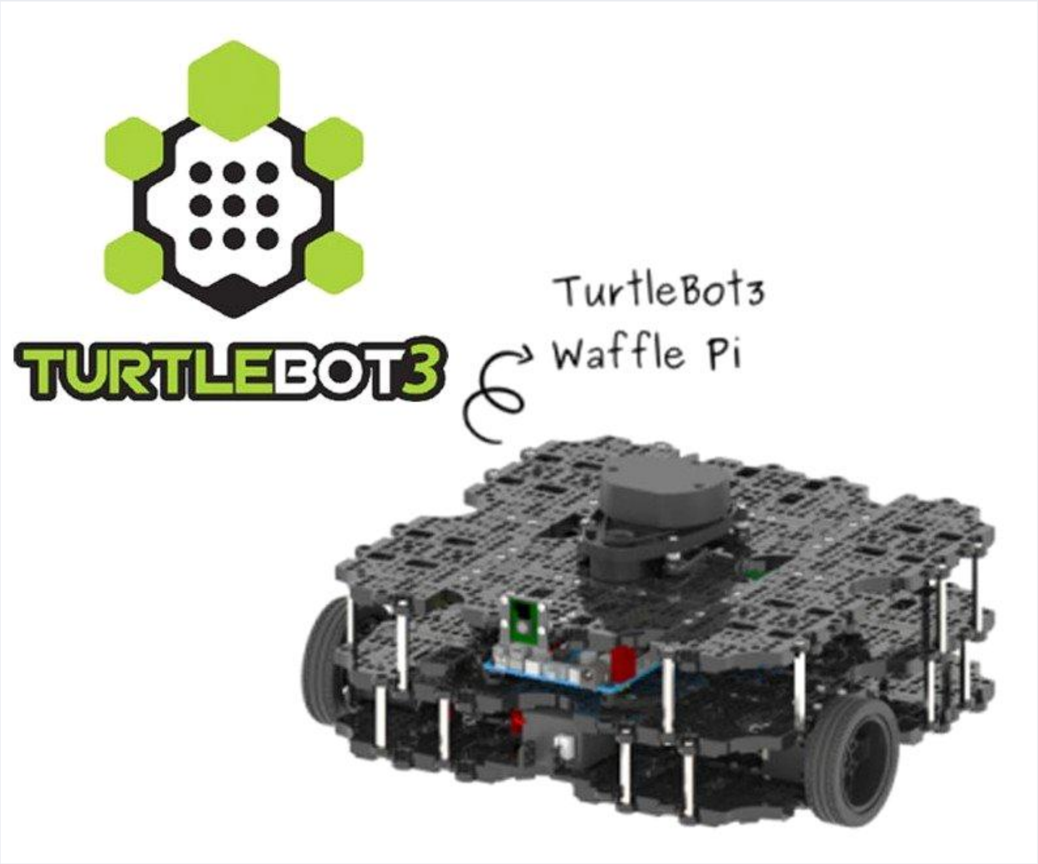
Students:
Filippo Samorì
Vittorio Caputo
Matteo Bonucci

Academic year 2024/2025

OBJECTIVE

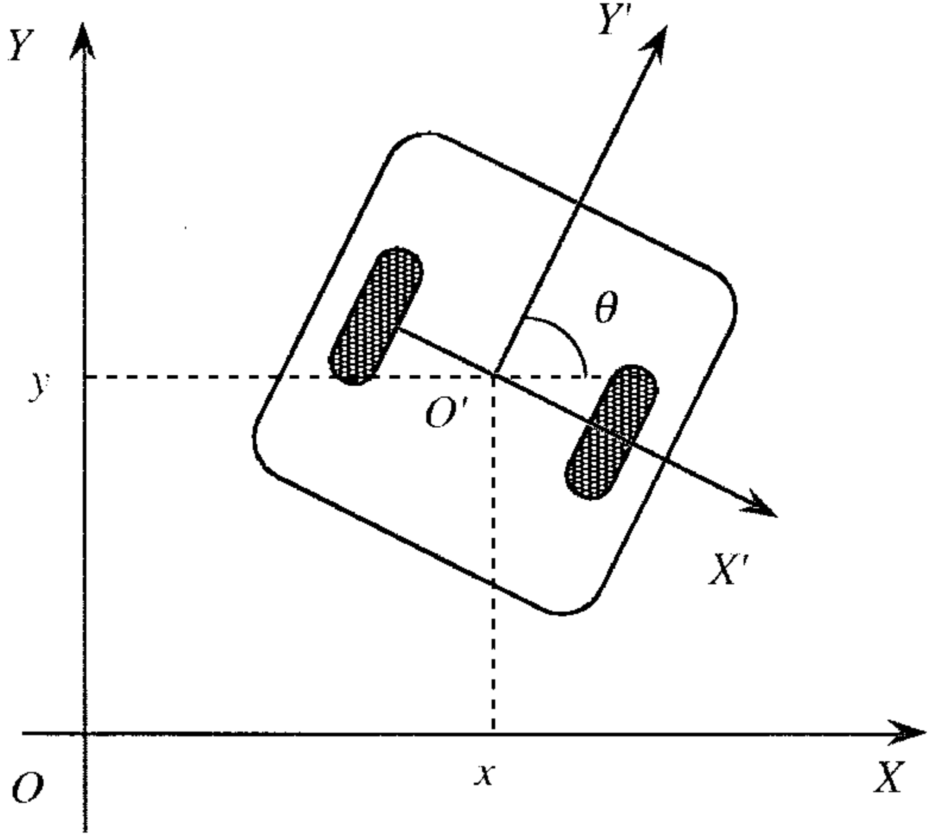
- In this project, we developed a simulation in **Gazebo** integrated with **ROS 2**, where a **TurtleBot** autonomously navigates in an unknown environment.

- The robot **autonomously explores** the environment to **detect**, **classify** and **estimate** the positions of three human targets.



MODEL OF THE ROBOT

- TurtleBot 3 **Waffle**
- **Unicycle** model with **differential** drive
- Equipped with **RGB-D** camera and **LIDAR**
- Configuration variables $\boldsymbol{q} = [\boldsymbol{x_r}, \boldsymbol{y_r}, \boldsymbol{\theta}]^{\boldsymbol{T}}$
- x_r and y_r are the coordinates of the center of the robot
- θ is the angle that describes the orientation of the robot



SENSORS: CAMERA PROPERTIES

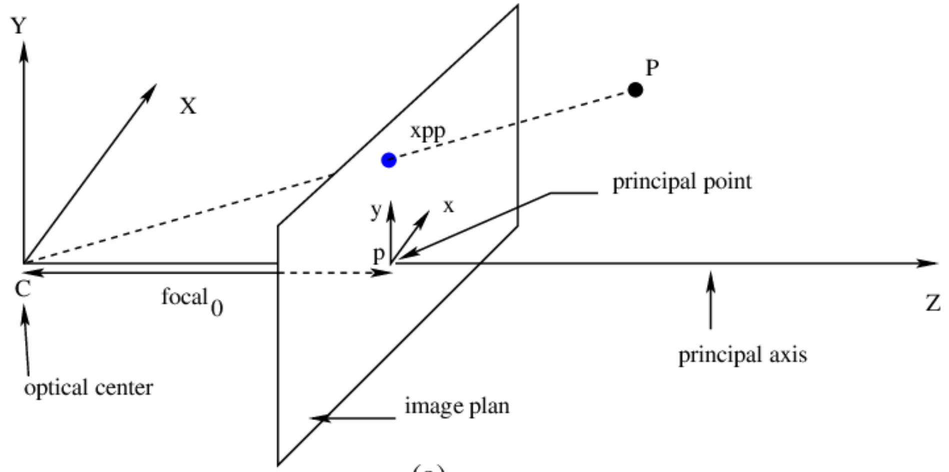
- Type: **RGB-D** – It provides both a RGB image and a DEPTH one
- **Update Rate**: 30 Hz
- **Field of View** (FOV): 1.02974 rad
- **Resolution**: 1920 x 1080 pixels
- **Clipping planes**:
  - **Near**: 0.1 m - Objects closer than 0.1 meters are ignored
  - **Far**: 10.0 m - Objects farther than 10 meters are ignored

SENSORS: CAMERA MODEL

The camera is modelled with the **Perspective Projection** model. Where the **3D** points in the camera reference frame are mapped in a **2D** point in the pixel image coordinates

- $\Delta \boldsymbol{u}$ is the pixel size along the x axis
- $\Delta \boldsymbol{v}$ is the pixel size along the y axis
- $\boldsymbol{u_0}$ is the x pixel offset between the center of the image and the left upper corner
- $\boldsymbol{v_0}$ is the y pixel offset between the center of the image and the left upper corner
- $\boldsymbol{f}$ is the focal length of the camera.

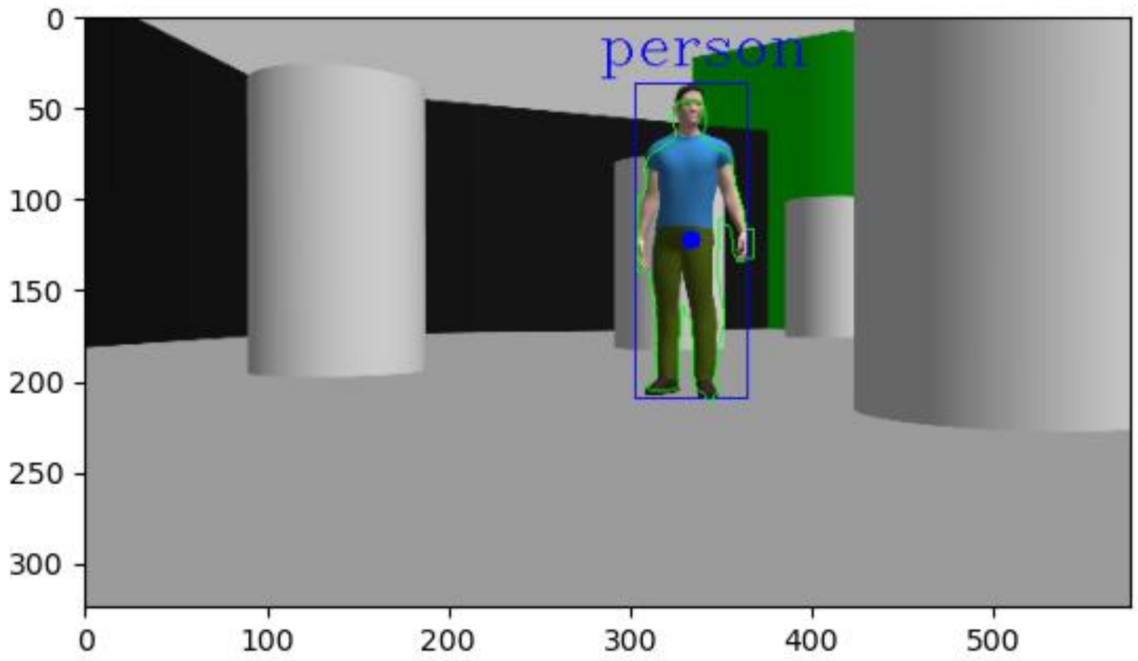
$$z\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{f}{\Delta u} & 0 & u_0 & 0 \\ 0 & \frac{f}{\Delta v} & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$



(a)

YOLO NEURAL NETWORK: TARGET DETECTION AND CLASSIFICATION

- The **YOLOv11-seg** Neural Network correctly **detects** and **classifies** the targets as a **person**
- It returns also the **bounding box** and the **mask**
- It is suitable for real-time applications



TARGET POSITION ESTIMATION

- First of all, after target detection, the **PPM** is used to map target **2D image pixel** (the center of the bounding box) in **3D CRF** coordinates.
- Then, the **target** position in **WRF** is estimated by knowing the **robot pose** and the **target** position in **CRF**
- The **target position** in WRF is **updated** with an **EKF** algorithm
- The depth z is computed as the median of the measurements inside the YOLO mask of the target

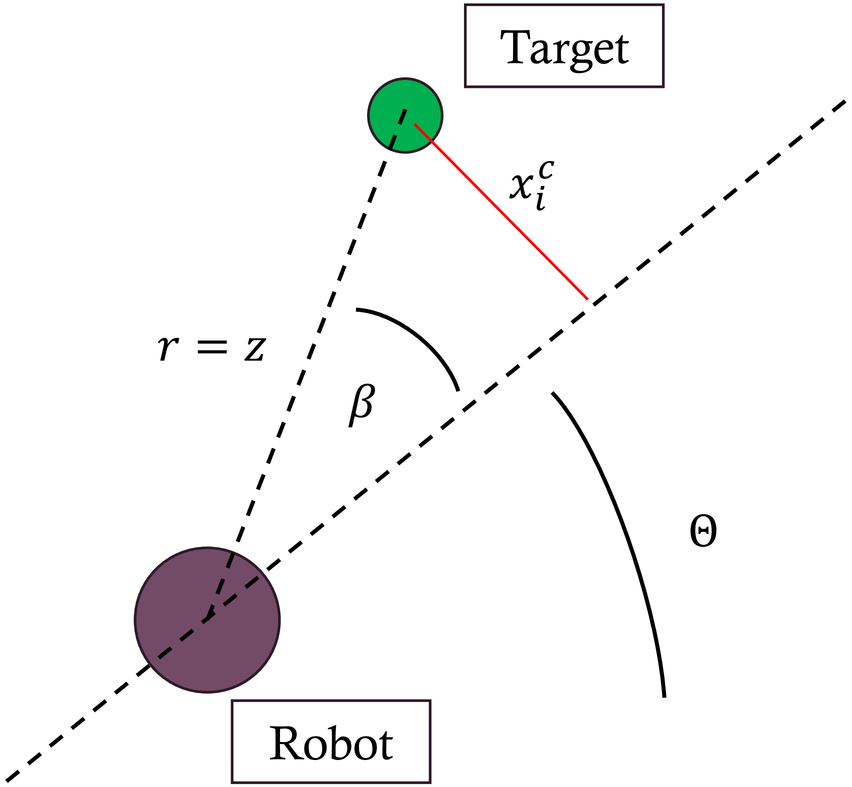
$$x_i^w = x_r^w + \mathrm{z}\cos\left(\theta + \sin^{-1}\left(\frac{-x_i^c}{z}\right)\right)$$

$$y_i^w = x_r^w + \mathrm{z}\sin\left(\theta + \sin^{-1}\left(\frac{-x_i^c}{z}\right)\right)$$

$$x_i^c = \frac{\tilde{u}}{f}\Delta u * z \qquad \tilde{u} = u - u_0$$

$(x_t\, , y_t\,)$ is the target position in **WRF**, X is the target x coordinate in the **CRF**

TARGET POSITION ESTIMATION



$$\begin{bmatrix} r \\ \beta \end{bmatrix} = \begin{bmatrix} \sqrt{(x_i^w - x^w)^2 + (y_i^w - y^w)^2} \\ \arctan(y_i - y^w)/(x_i^w - x^w) - \theta \end{bmatrix}$$

$$\beta = \arcsin(-\frac{x_i^c}{z}) = \arcsin(-\frac{\tilde{u}}{C_u})$$

$$\begin{bmatrix} z \\ \tilde{u} \end{bmatrix} = h(q, p) = \begin{bmatrix} \sqrt{(x_i^w - x^w)^2 + (y_i^w - y^w)^2} \\ \sin(\arctan(y_i^w - y^w)/(x_i^w - x^w) - \theta)C_u \end{bmatrix}$$

EXTENDED KALMAN FILTER

The **Jacobian** of the measurement model with respect to the target position is the following:

$$H = \begin{bmatrix} -\cos(\arctan(y_i^w - y^w)/(x_i^w - x^w) - \theta))^{\frac{(x_i^w - x^w)}{z}} \frac{(-y_i^w + y^w)}{z^2} C_u & -\cos(\arctan(y_i^w - y^w)/(x_i^w - x^w) - \theta))^{\frac{(y_i^w - y^w)}{z}} \frac{(x_i^w - x^w)}{z^2} C_u \end{bmatrix}$$

The model "dynamics" of the target is the following:

$$\tilde{p}(k+1) = p(K)$$

$$\tilde{P}(k+1) = P(K) \qquad covariance$$

then the estimated new pose $\tilde{p}$ can be updated using the sensor information and the Kalman gain:

$$v = measurement - h(q, \tilde{p}) \qquad innovation$$

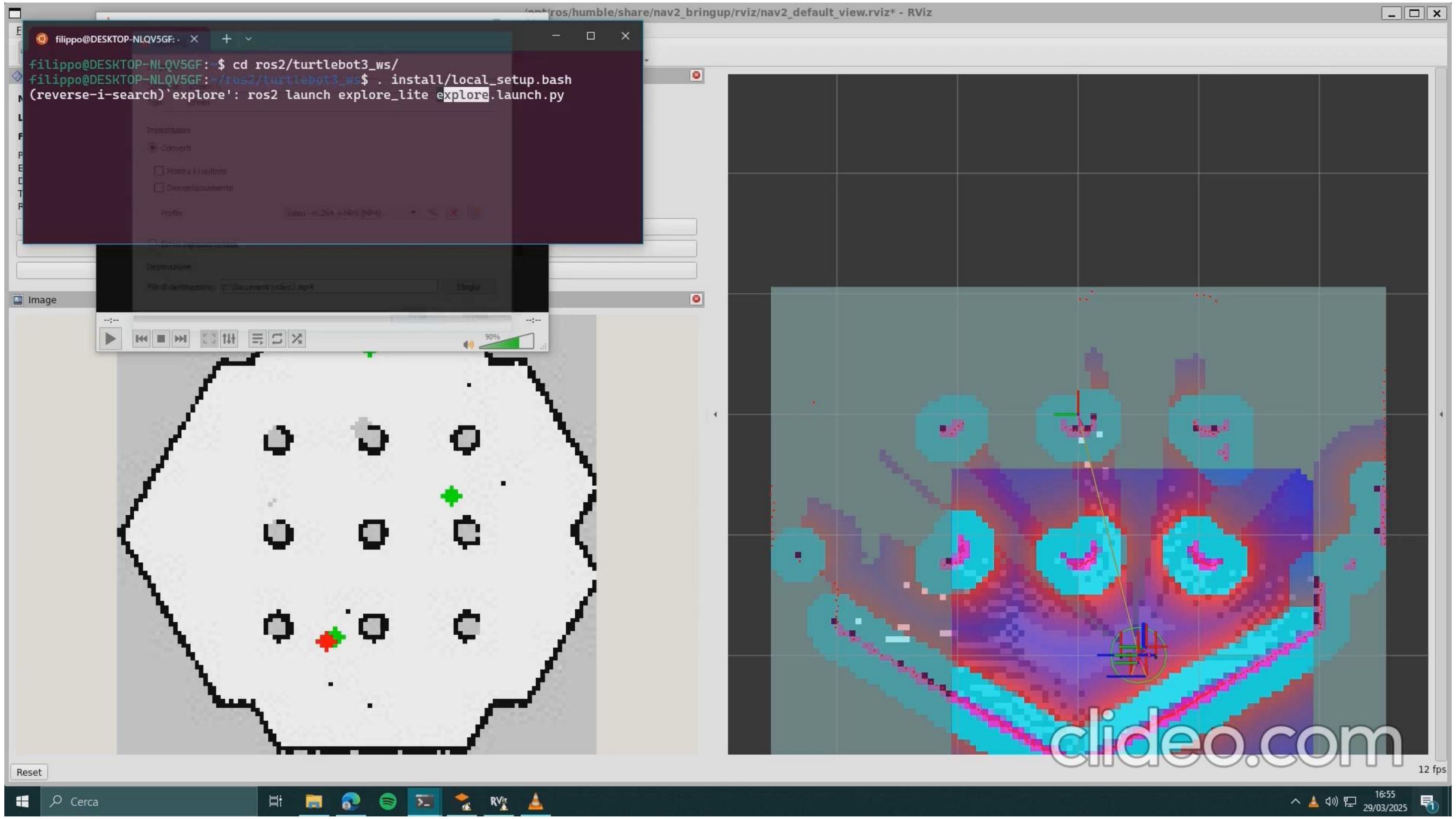
$$K = PH^T(HPH^T + W)^{-1} \qquad Kalman\ \ gain$$

$$p(k) = \tilde{p}(k) + Kv$$

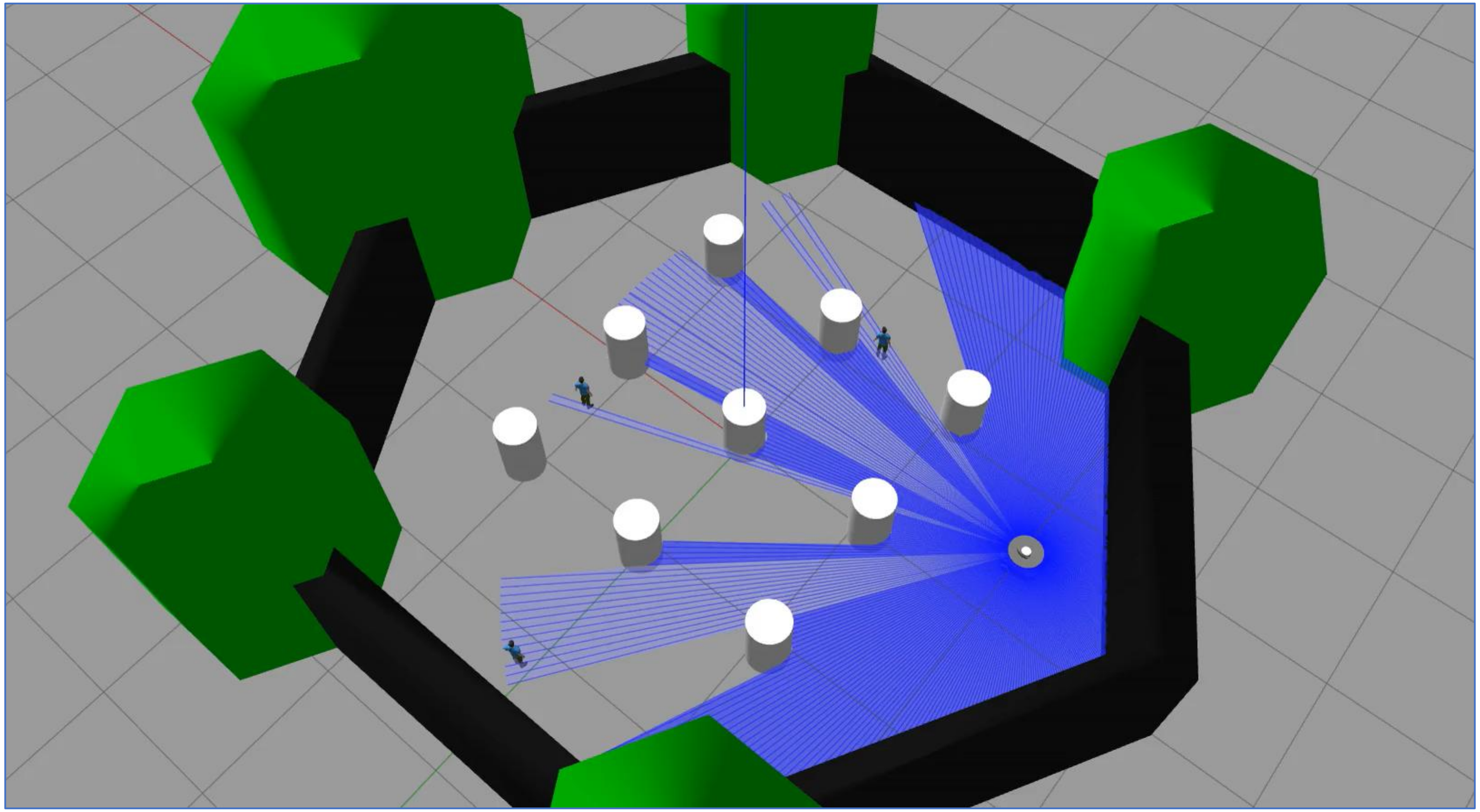
$$P(k) = \tilde{P}(k) - KH\tilde{P}(k)$$

Note: This model assumes the robot configuration to be perfectly known

TARGET POSITION ESTIMATION



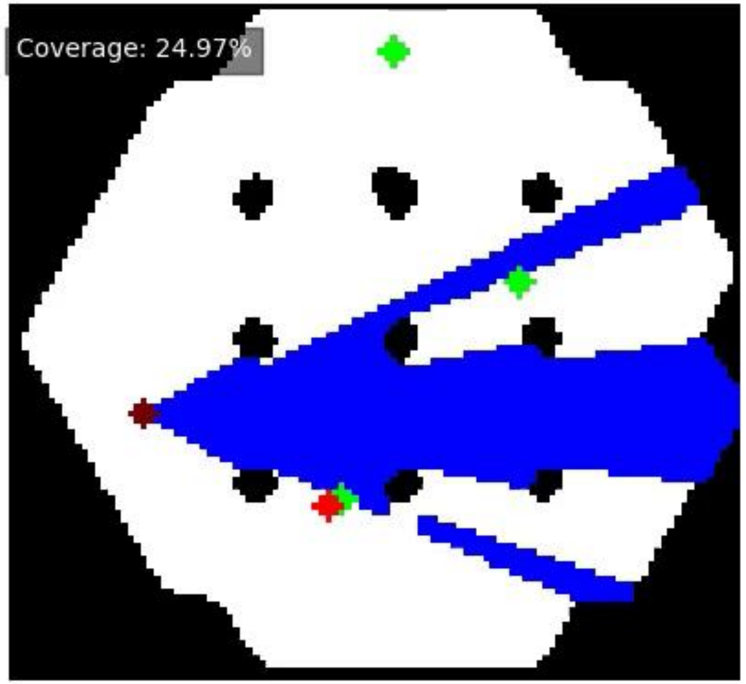
ENVIRONMENT EXPLORATION



AREA COVERAGE

- The **FOV** is represented as python **set of pixels** in the map, accounting for obstacles that block visibility
- During the exploration:
  - A new FOV set is computed at each step
  - It is incrementally merged with the previous FOV set using a **set union** operation
- The **total covered area** by the robot is the count of pixels in the final FOV set
- The **area coverage** is calculated as:

$$Area\ Coverage = \frac{Pixels\ in\ total\ FOV}{Pixels\ in\ Region\ of\ Interest\ (ROI)}$$



RESULTS

- **Error distances** between the true targets and the estimated ones : [0.11, 0.08, 0.04]
- **Estimated poses**: [-0.371, -1.097], [0.086, 2.188], [1.040, 0.503]
- **Covariance matrices**:

$$\begin{bmatrix} 0.00011209 & -0.0003046 \\ -0.0003046 & 0.0008417 \end{bmatrix};\begin{bmatrix} 5.0911e{-07} & -1.3058e{-07} \\ -1.3058e{-07} & 6.6177e{-07} \end{bmatrix};\begin{bmatrix} 2.9291e{-06} & -7.7388e{-07} \\ -7.7388e{-07} & 9.1198e{-07} \end{bmatrix}$$

- **Area Coverage**: 99%
- Our model is able to detect targets within the environment with **good accuracy**, as shown by the error.
- The mathematical model used to estimate the target's position captures its dependence on the camera state (position), while the neural network provides **robustness** to changes in the target and image conditions.

THANK YOU FOR YOUR ATTENTION