Master of Science in Artificial Intelligence and Robotics

Safe robot navigation in a crowd: Application to the TIAGo mobile manipulator

SAPIENZA
UNIVERSITÀ DI ROMA

Candidate

Giovanbattista Gravina
ID number 2021526

Thesis Advisor

Prof. Giuseppe Oriolo

Introduction

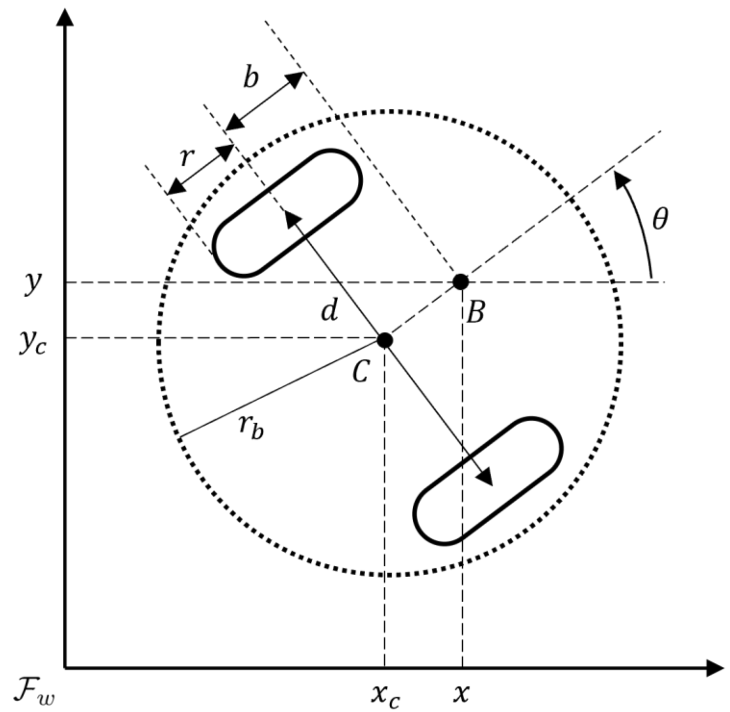
Problem: safe TIAGo robot navigation in a crowded environment

Methodologies:

- laser-based framework
- Nonlinear Model Predictive Control (NMPC) algorithm
- Control Barrier Functions (CBFs) for collision avoidance

Problem formulation
Differential-drive robot model

$$\dot{\boldsymbol{x}} = \boldsymbol{\phi}(\boldsymbol{x}, \boldsymbol{u}) = \begin{pmatrix} v\cos\theta - \omega b \sin\theta \\ v\sin\theta + \omega b\cos\theta \\ \omega \\ \frac{r}{2}(\dot{\omega}^R + \dot{\omega}^L) \\ \frac{r}{d}(\dot{\omega}^R - \dot{\omega}^L) \end{pmatrix}$$



with

- $\boldsymbol{x} = (\boldsymbol{q}, \boldsymbol{\nu}) \in \mathbb{R}^5$ robot state
- $\boldsymbol{q} = (x, y, \theta)$ robot configuration
- $\boldsymbol{\nu} = (v, \omega)$ robot pseudovelocities
- $\boldsymbol{u} = (\dot{\omega}^R, \dot{\omega}^L)$ control inputs

Problem formulation

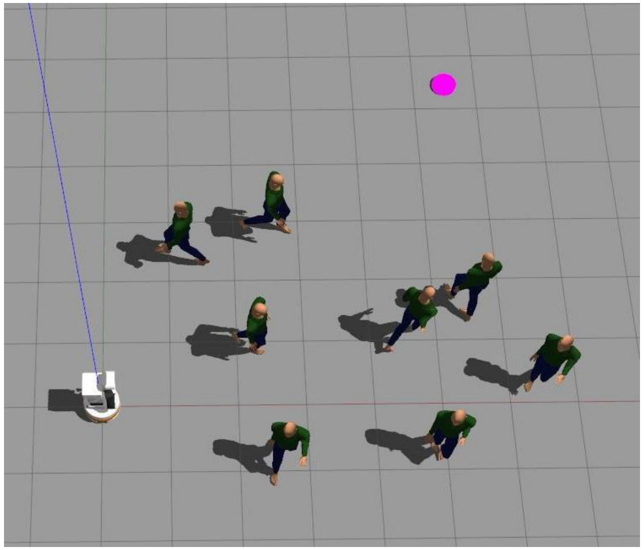
Navigation task

Given

- a workspace populated by moving agents
- a desired goal position $\boldsymbol{p}_g = (x_g, y_g)$
- observations gathered by laser

generate a robot motion that

- is consistent with the robot model
- respects state and control bounds
- avoids collision with agents
- reaches the goal position $\boldsymbol{p}_g$



Adopted framework

Two-module scheme

Assuming that the robot is localized, at time instant t_k:

- crowd prediction module

  receives sensor data $\mathcal{S}_k$ and robot state $\boldsymbol{x}_k$

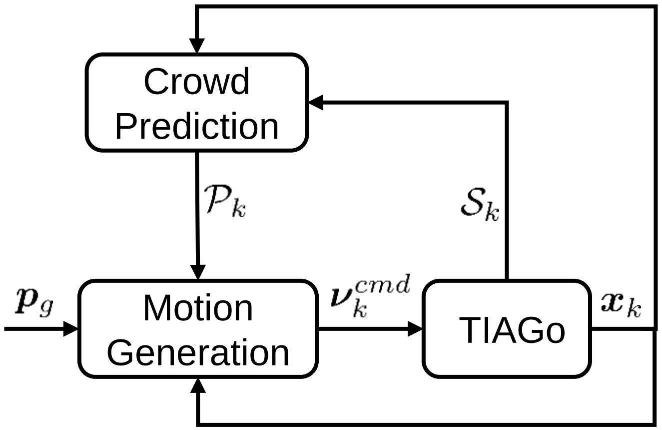
  generates the predicted agents' motion $\mathcal{P}_k$

- motion generation module

  generates real-time control inputs $\boldsymbol{u}_k$ to

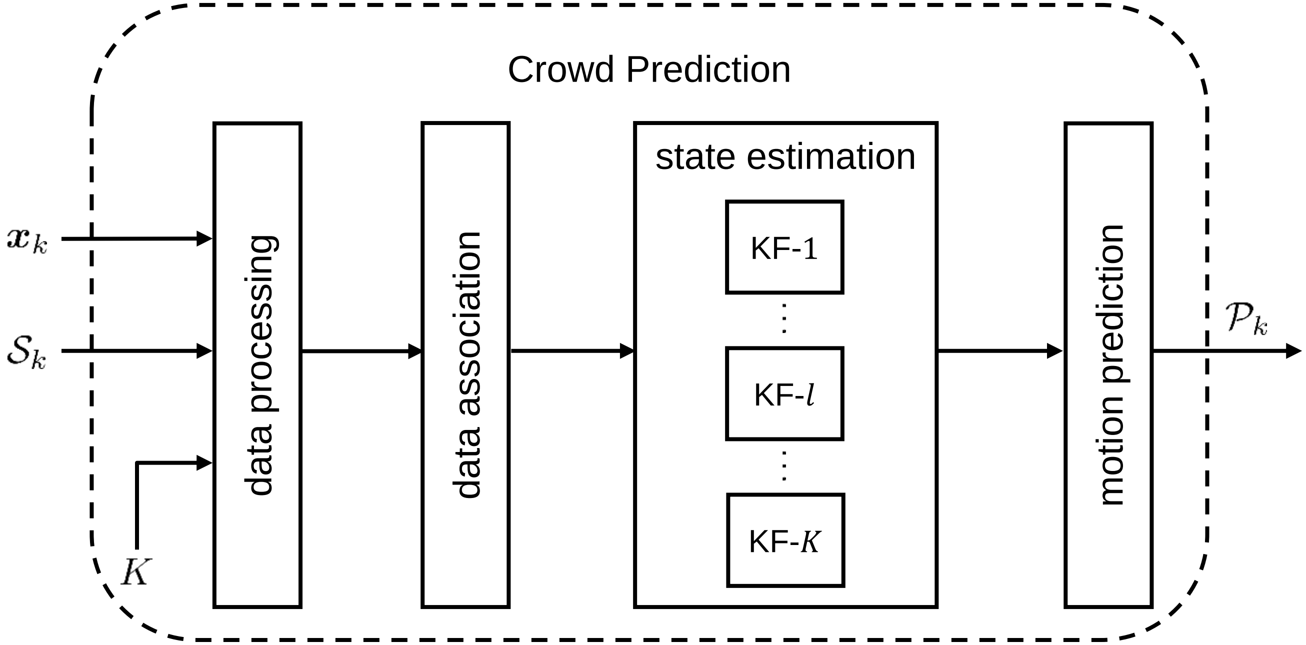
  – accomplish the navigation task

  – keep the robot inside an admissible region



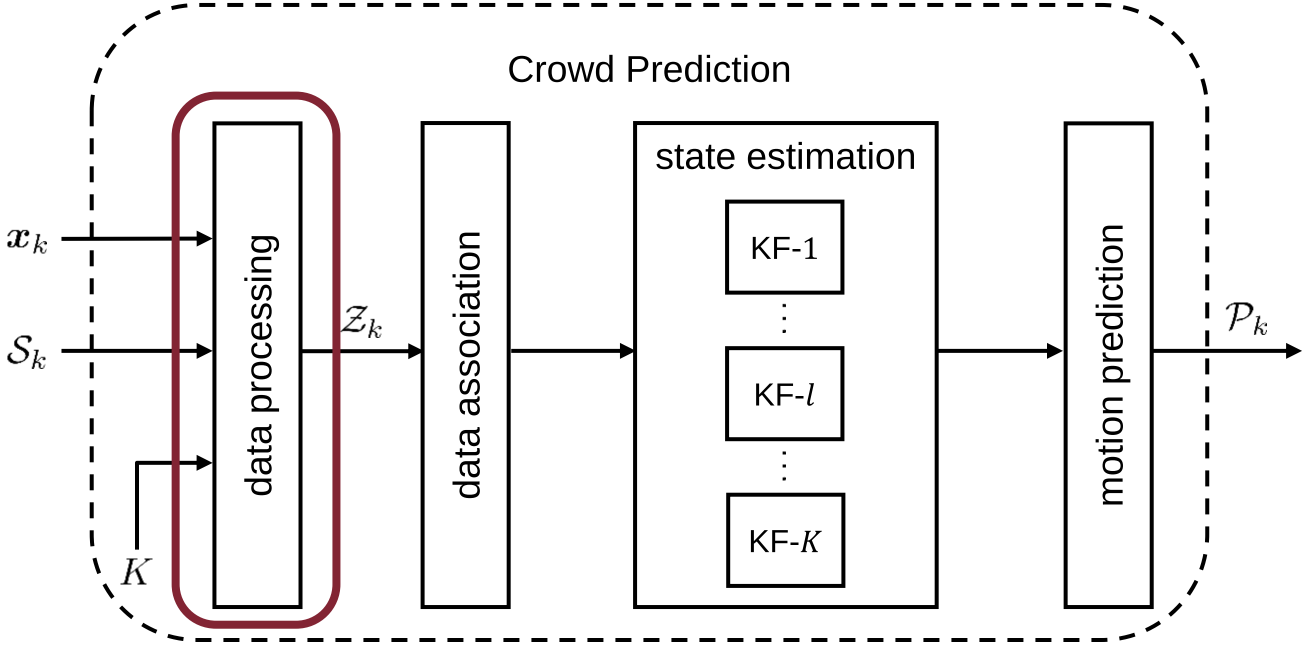
Note: $\boldsymbol{u}_k$ is integrated to obtain the admissible robot commands $\boldsymbol{\nu}_k^{cmd}$

Crowd prediction module
Conceptual scheme



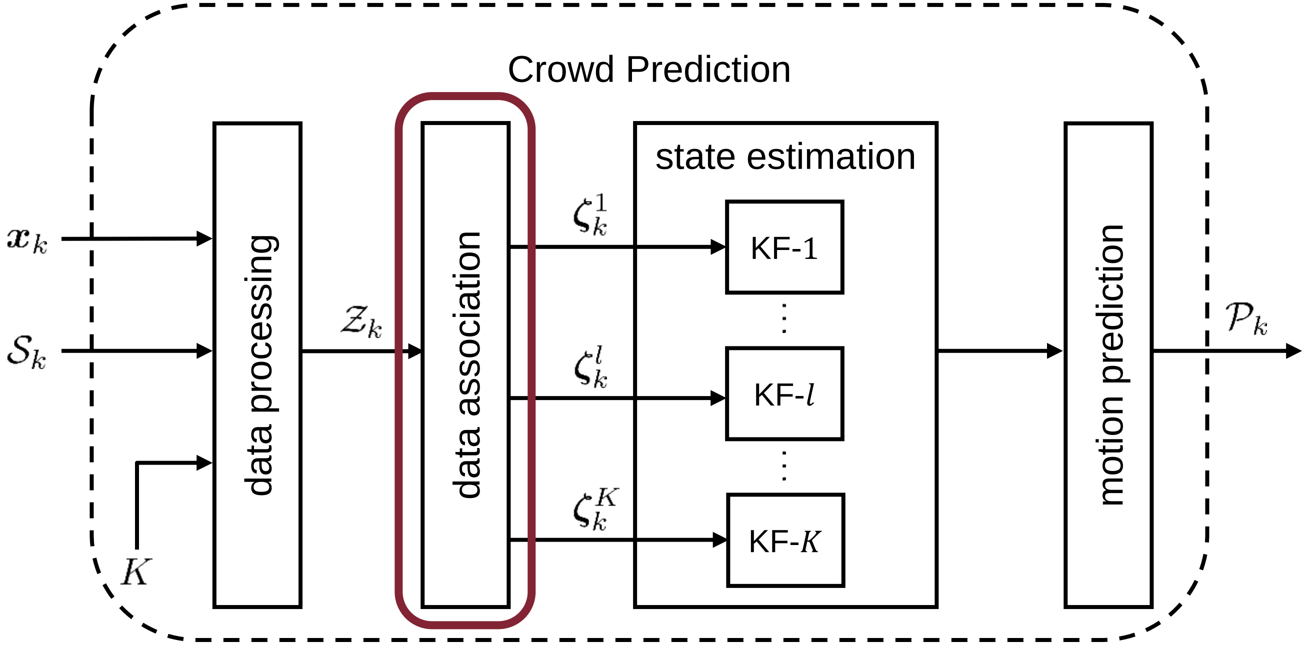
where K Kalman Filters (KFs) are employed, with K a user-specified number

Crowd prediction module
Data processing



- identify N_c clusters
- for each cluster, compute the average point as its representative point
- collect the $M = \min(N_c, K)$ points closest to the robot in $\mathcal{Z}_k = \{\boldsymbol{z}_k^1, \ldots, \boldsymbol{z}_k^M\}$

Crowd prediction module
Data association

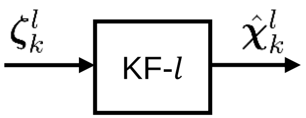


The representative point associated with the KF-l is denoted by ζ_k^l

Note: if $M < K$, $\zeta = \emptyset$ is assigned to the remaining $K - M$ KFs

Crowd prediction module
State estimation

State of the l-th agent at t_k: $\boldsymbol{\chi}_k^l = (\boldsymbol{p}_k^l, \dot{\boldsymbol{p}}_k^l)$



with $\boldsymbol{p}_k^l, \dot{\boldsymbol{p}}_k^l$ the position and velocity of its representative point

State-transition and output models are

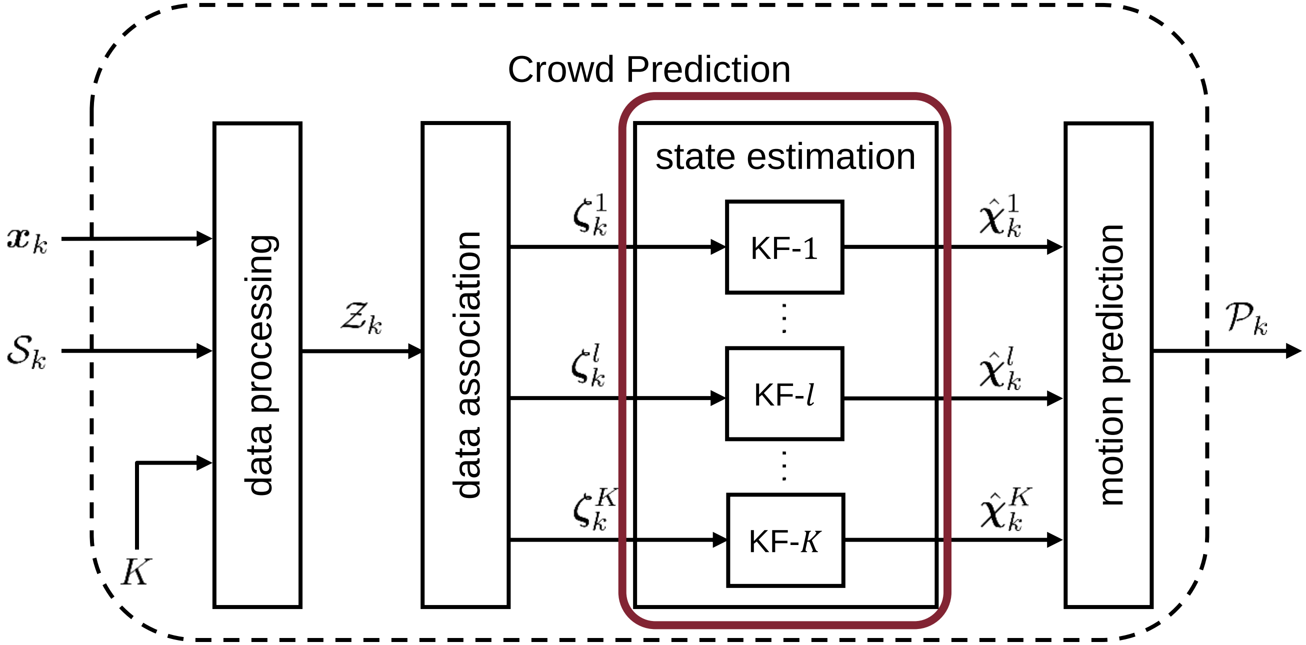
$$\boldsymbol{\chi}_{k+1}^l = \underbrace{\begin{pmatrix} \boldsymbol{I}_2 & \delta \boldsymbol{I}_2 \\ \boldsymbol{0}_{2\times 2} & \boldsymbol{I}_2 \end{pmatrix}}_{\boldsymbol{A}} \boldsymbol{\chi}_k^l + \boldsymbol{v}_k$$

$$\zeta_k^l = \underbrace{\begin{pmatrix} \boldsymbol{I}_2 & \boldsymbol{0}_{2\times 2} \end{pmatrix}}_{\boldsymbol{C}} \boldsymbol{\chi}_k^l + \boldsymbol{w}_k$$

with $\boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{V}_k), \; \boldsymbol{w}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{W}_k)$

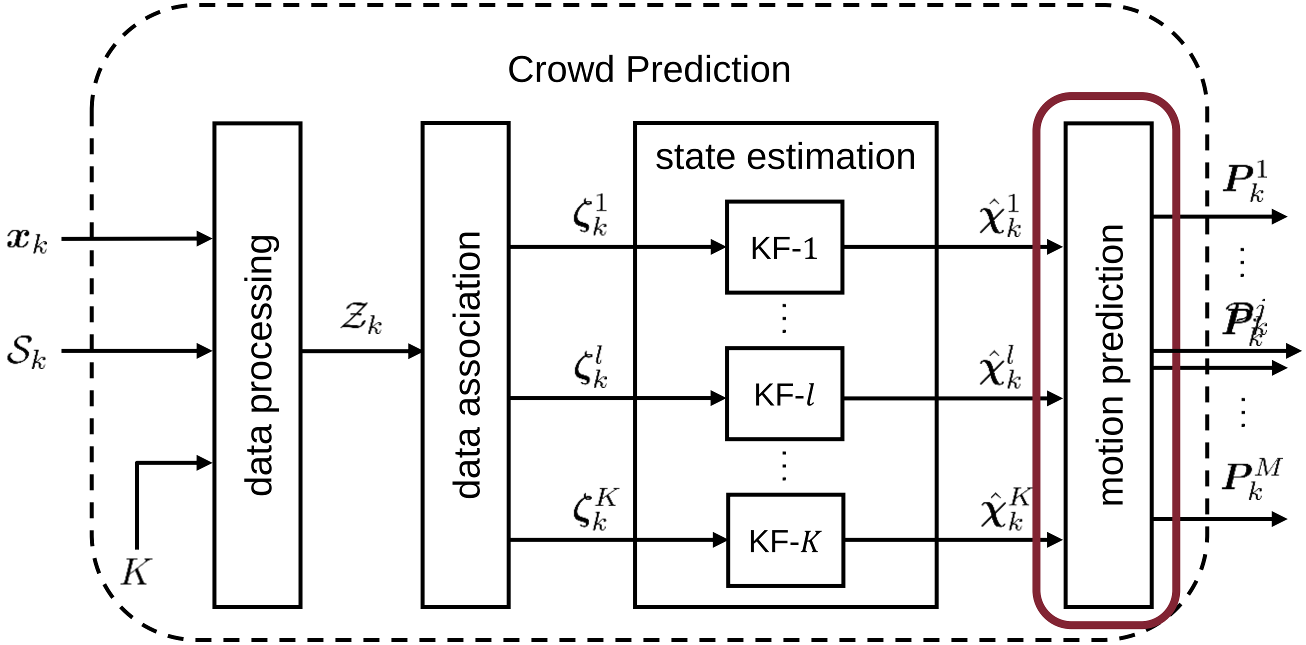
The KF-l produces $\hat{\boldsymbol{\chi}}_k^l = (\hat{\boldsymbol{p}}_k^l, \hat{\dot{\boldsymbol{p}}}_k^l)$, i.e., the estimate of the l-th agent's state

Crowd prediction module
State estimation



Crowd prediction module

Motion prediction



compute the set of predicted motion of the agents $\mathcal{P}_k = \{\boldsymbol{P}_k^1, \dots, \boldsymbol{P}_k^M\}$

where $\boldsymbol{P}_k^j = (\boldsymbol{p}_{0|k}^j, \dots, \boldsymbol{p}_{N|k}^j)$, with N the number of sampling intervals and $\boldsymbol{p}_{i|k}^j$ the predicted position of the j-th agent at t_{k+i}

Motion generation module

NMPC algorithm

Generate $\boldsymbol{u}_k = (\dot{\omega}_k^R, \dot{\omega}_k^L)$ solving a NonLinear Programming problem (NLP)

- decision variables: $\boldsymbol{X}_k = (\boldsymbol{x}_{0|k}, \ldots, \boldsymbol{x}_{N|k}),\ \boldsymbol{U}_k = (\boldsymbol{u}_{0|k}, \ldots, \boldsymbol{u}_{N-1|k})$

with $\boldsymbol{x}_{i|k}, \boldsymbol{u}_{i|k}$ the robot states and control inputs at t_{k+i}

- running and terminal cost:

$$V_{i|k}(\boldsymbol{x}_{i|k}, \boldsymbol{u}_{i|k}) = \boldsymbol{e}_{i|k}^T \boldsymbol{Q} \boldsymbol{e}_{i|k} + \boldsymbol{\nu}_{i|k}^T \boldsymbol{R} \boldsymbol{\nu}_{i|k} + \boldsymbol{u}_{i|k}^T \boldsymbol{S} \boldsymbol{u}_{i|k},$$

$$V_{N|k}(\boldsymbol{x}_{N|k}) = \boldsymbol{e}_{N|k}^T \boldsymbol{Q}_N \boldsymbol{e}_{N|k} + \boldsymbol{\nu}_{N|k}^T \boldsymbol{R}_N \boldsymbol{\nu}_{N|k}$$

with $\boldsymbol{e}_{i|k} = \boldsymbol{p}_g - \boldsymbol{\eta}_{i|k},\ \boldsymbol{\eta}_{i|k} = (x_{i|k}^{(1)}, x_{i|k}^{(2)}),\ \ \boldsymbol{\nu}_{i|k} = (x_{i|k}^{(4)}, x_{i|k}^{(5)})$ and

$\boldsymbol{Q}, \boldsymbol{R}, \boldsymbol{S}, \boldsymbol{Q}_N, \boldsymbol{R}_N$ weighting matrices of appropriate dimensions

Motion generation module
NMPC algorithm

The NLP is formulated as

$$\min_{\boldsymbol{X}_k, \boldsymbol{U}_k} \sum_{i=0}^{N-1} V_{i|k}(\boldsymbol{x}_{i|k}, \boldsymbol{u}_{i|k}) + V_{N|k}(\boldsymbol{x}_{N|k})$$

subject to:

$$\boldsymbol{x}_{0|k} - \boldsymbol{x}_k = \boldsymbol{0}$$

$$\boldsymbol{x}_{i+1|k} - \boldsymbol{\phi}_{\text{d-t}}(\boldsymbol{x}_{i|k}, \boldsymbol{u}_{i|k}) = \boldsymbol{0} \qquad i = 0, \dots, N-1$$

$$\underline{\boldsymbol{g}} \leq \boldsymbol{C}\boldsymbol{x}_{i|k} \leq \overline{\boldsymbol{g}} \qquad i = 1, \dots, N$$

$$\underline{\boldsymbol{l}} \leq \boldsymbol{D}\boldsymbol{u}_{i|k} \leq \overline{\boldsymbol{l}} \qquad i = 0, \dots, N-1$$

$$\textit{collision avoidance constraints at } t_k, \dots, t_{k+N-1}$$

imposes the initial state at the current robot state

enforces the system dynamics

constraints on the robot states and on the control inputs

Motion generation module

Collision avoidance constraints

Define the set containing all the collision-free robot states as a *safe-set*

$$\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^5 : h(\boldsymbol{x}) \geq 0\}$$

with $h : \mathbb{R}^5 \mapsto \mathbb{R}$ a continuous function

- the system is *safe* if $\mathcal{C}$ is forward-invariant, i.e., $\boldsymbol{x}_0 \in \mathcal{C} \implies \boldsymbol{x}_k \in \mathcal{C} \ \forall k \in \mathbb{N}$
- the function h is a discrete-time CBF on $\mathcal{C}$ if it satisfies

$$h(\boldsymbol{x}_{k+1}) \geq (1-\gamma)h(\boldsymbol{x}_k), \quad 0 < \gamma \leq 1$$

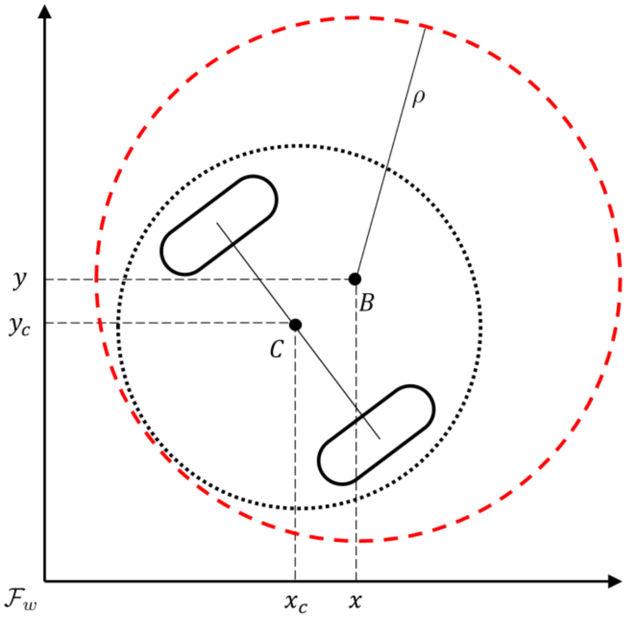
If h is a discrete-time CBF the set $\mathcal{C}$ is forward-invariant, thus the system is safe

- constraint $h(\boldsymbol{x}) \geq 0$ is only active on $\partial\mathcal{C} = \{\boldsymbol{x} \in \mathbb{R}^5 : h(\boldsymbol{x}) = 0\}$
- CBF-based constraint always affects the solution

Motion generation module
Collision avoidance constraints

Considering

- a bounding circle centered in B with radius ρ



Motion generation module

Collision avoidance constraints

Considering

- a bounding circle centered in B with radius ρ
- an m-sided convex polygon as admissible region
- a safety clearance $d_s > 0$ for each agent

the safe-set of robot states is given by

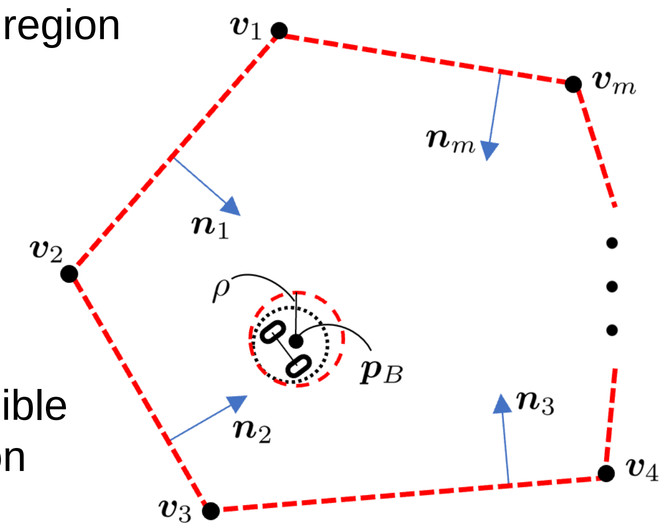
$$\overline{\mathcal{C}} = \{\boldsymbol{x} \in \mathbb{R}^5 : \overline{\boldsymbol{h}}(\boldsymbol{x}) \geq \mathbf{0}\}$$

with

$$\overline{\boldsymbol{h}}(\boldsymbol{x}) = \begin{bmatrix} \big(\boldsymbol{p}_B(\boldsymbol{x}) - \boldsymbol{v}_1\big) \cdot \boldsymbol{n}_1 - \rho \\ \vdots \\ \big(\boldsymbol{p}_B(\boldsymbol{x}) - \boldsymbol{v}_m\big) \cdot \boldsymbol{n}_m - \rho \\ \|\boldsymbol{p}_B(\boldsymbol{x}) - \boldsymbol{p}^1\|^2 - (\rho + d_s)^2 \\ \vdots \\ \|\boldsymbol{p}_B(\boldsymbol{x}) - \boldsymbol{p}^M\|^2 - (\rho + d_s)^2 \end{bmatrix}$$

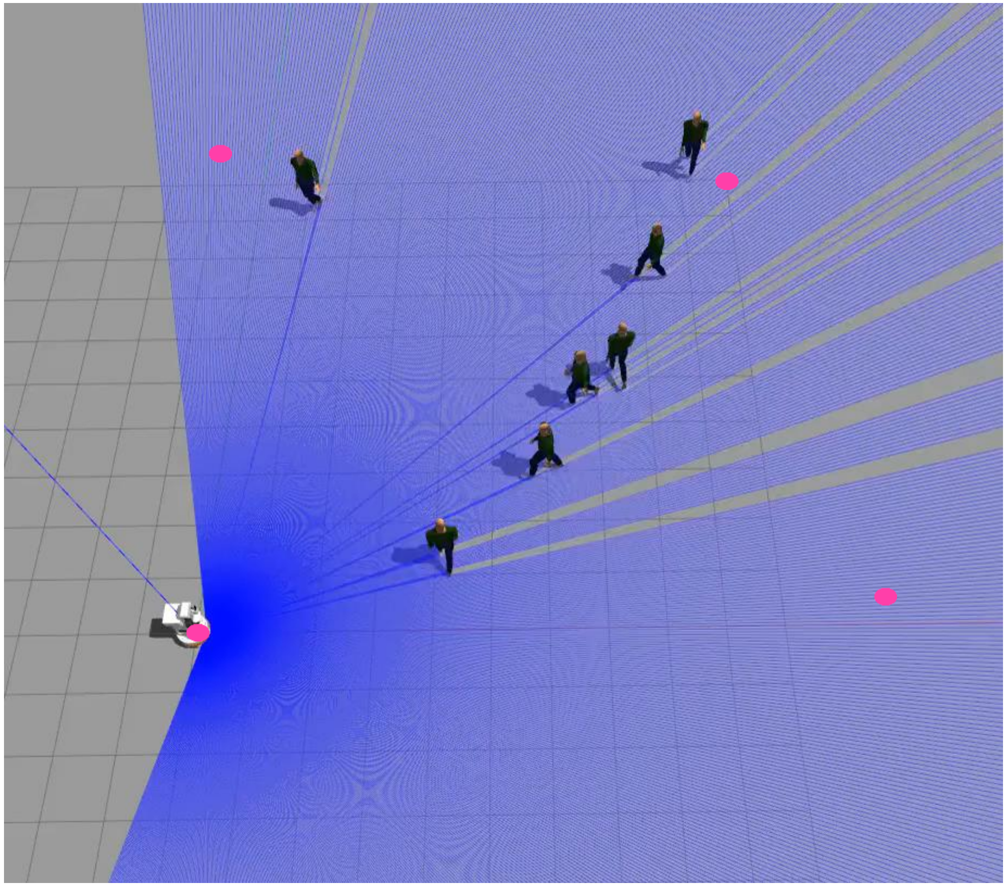
admissible region

agents



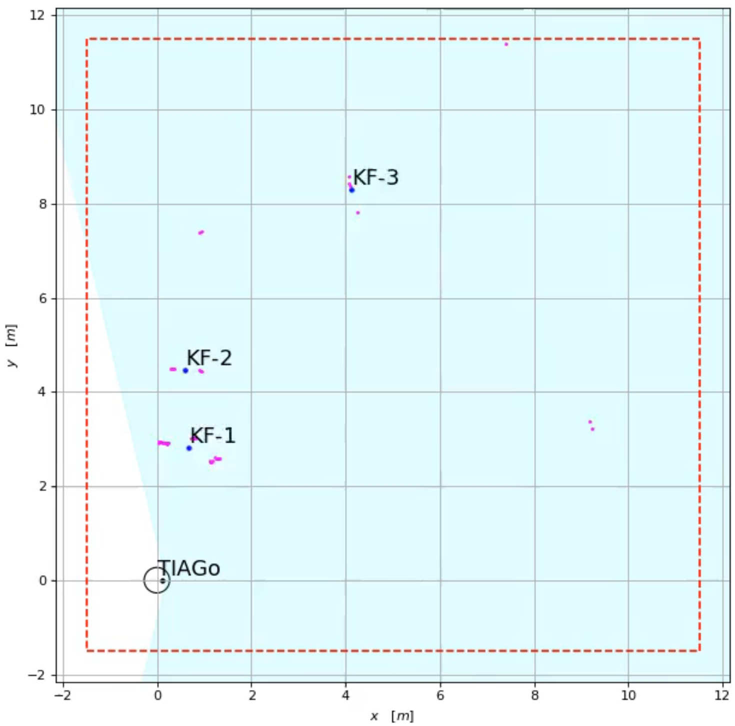
Simulation

- acados package
- Python code
- Gazebo simulator
- control frequency
  $f = 10\ [Hz]$
- $\delta = \frac{1}{f} = 0.1\ [s]$
- $N = 20$, thus
  $T = N\delta = 2\ [s]$

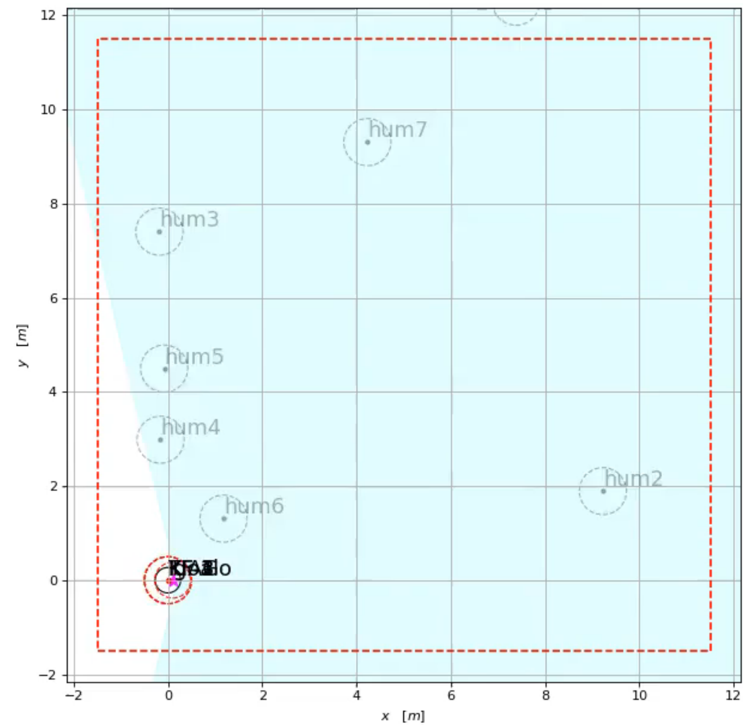


$2.5 \times$ speed

Simulation



Crowd prediction module



Motion generation module

2.5 × speed

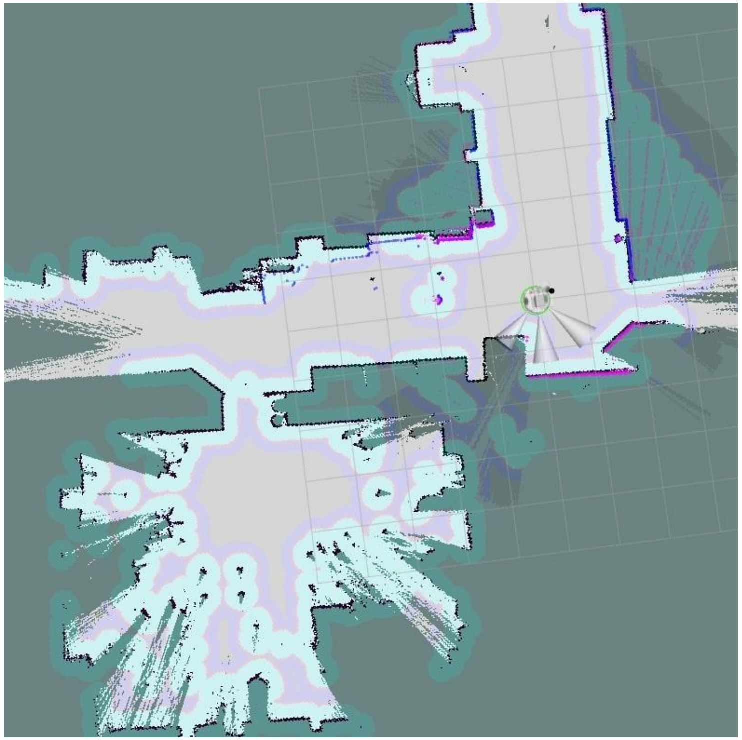
Experiments
Setup

Two scenarios

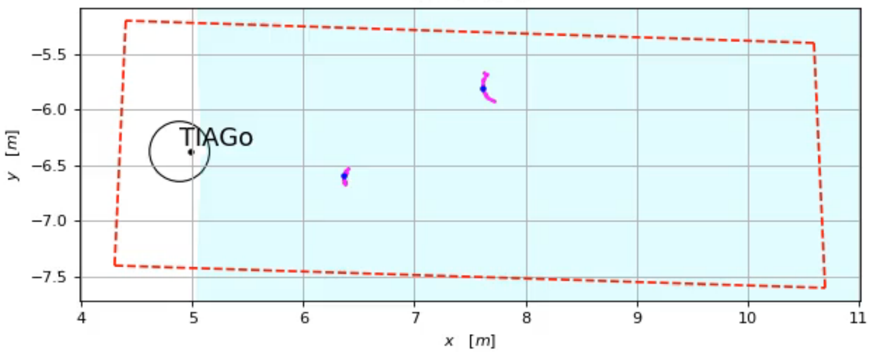
1. Static obstacles
2. Moving humans

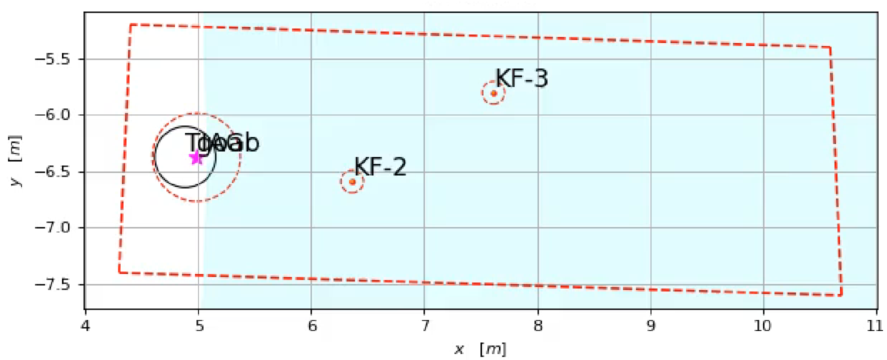
A map for robot localization is obtained

- from the laser scanner
- using *gmapping*



Experiments
Static obstacles

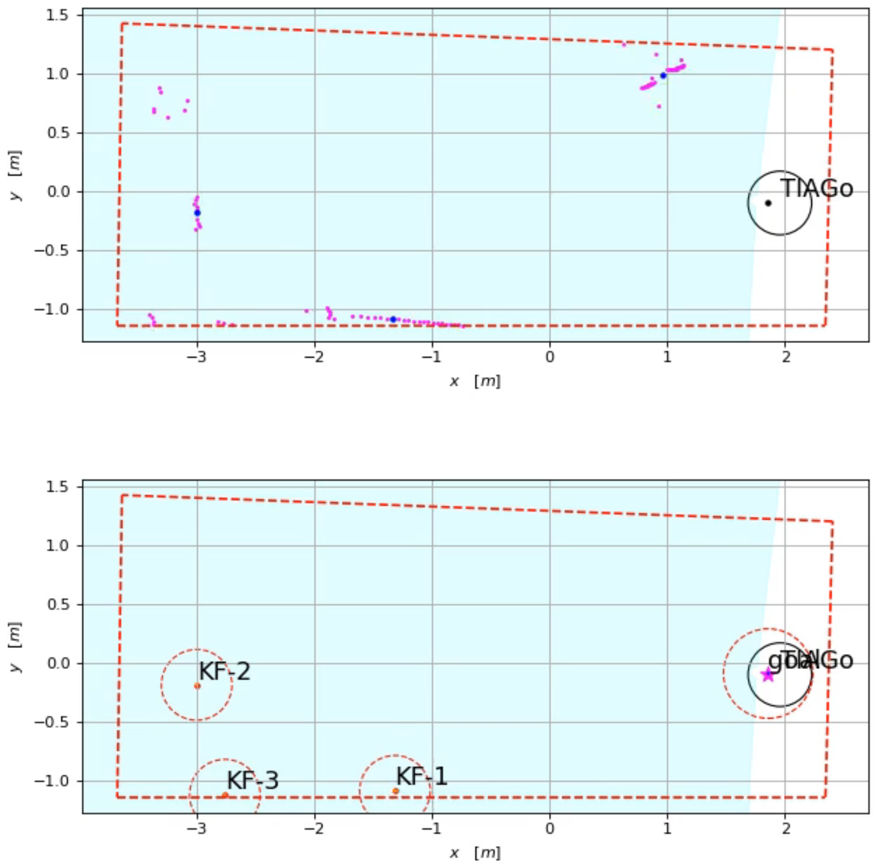


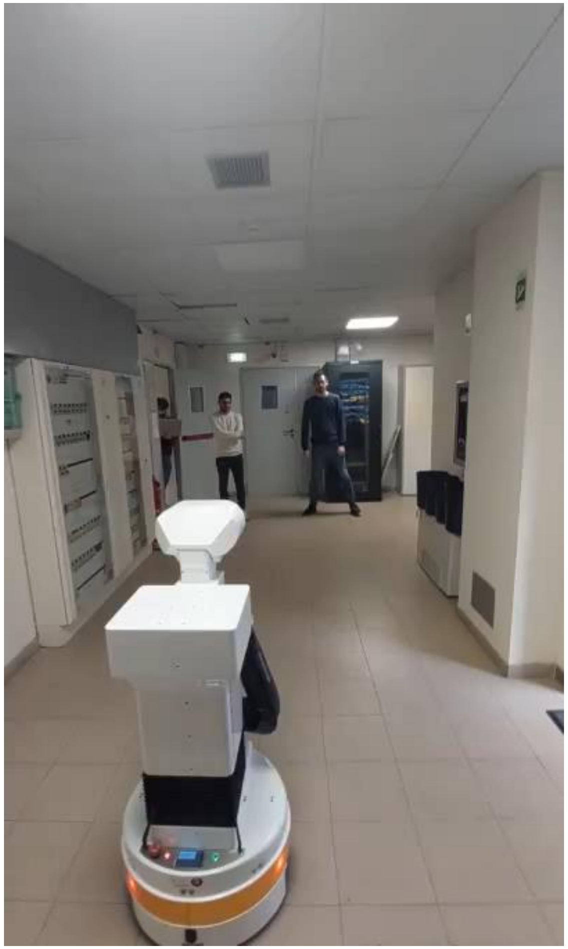


2.5 × speed

Experiments

Moving humans





Conclusions

- two-module framework for safe robot navigation in a crowd
- efficient combination of NMPC and CBFs for real-time collision avoidance
- pure laser-based perception results in poor and noisy data
- effectiveness of the average point as agent's representative point
- future research directions:
  - extensive experimental validation
  - exploit sensor fusion: combination of laser, ultrasound sonars and RGB-D camera
  - high-level online motion planning module for navigation in complex environments