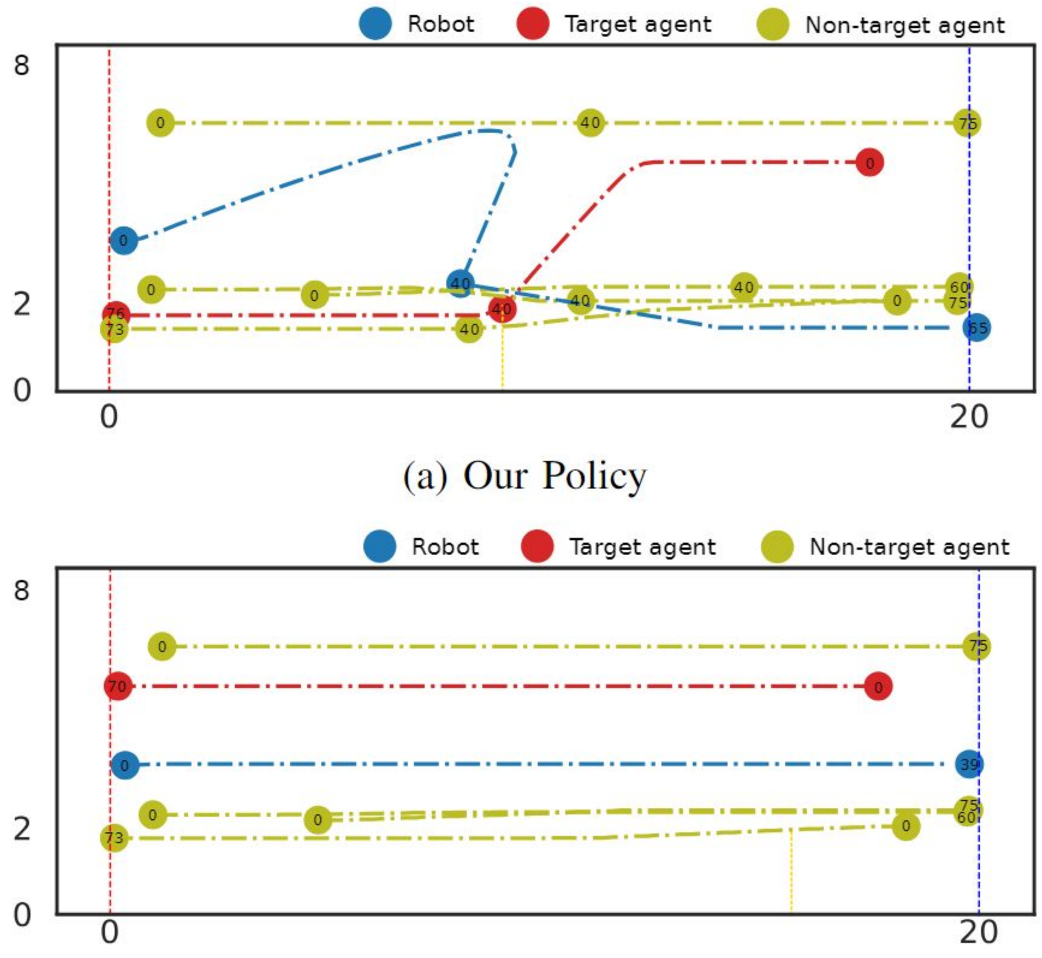
I-ORCA – Implicitly Nudging Human Trajectories

Shreyas Bhat, Kumar Akash

- Humans and robots are increasingly sharing the same physical space
- Past research has focused on designing collision-free trajectories for robots in crowded environments
- We envision a future where robots not only avoid collisions, but also actively guide them towards their objectives
- Such robots can help with crowd control, evacuation, tour guides, etc.



(a) Our Policy

(b) ORCA

Related Work

Social Navigation

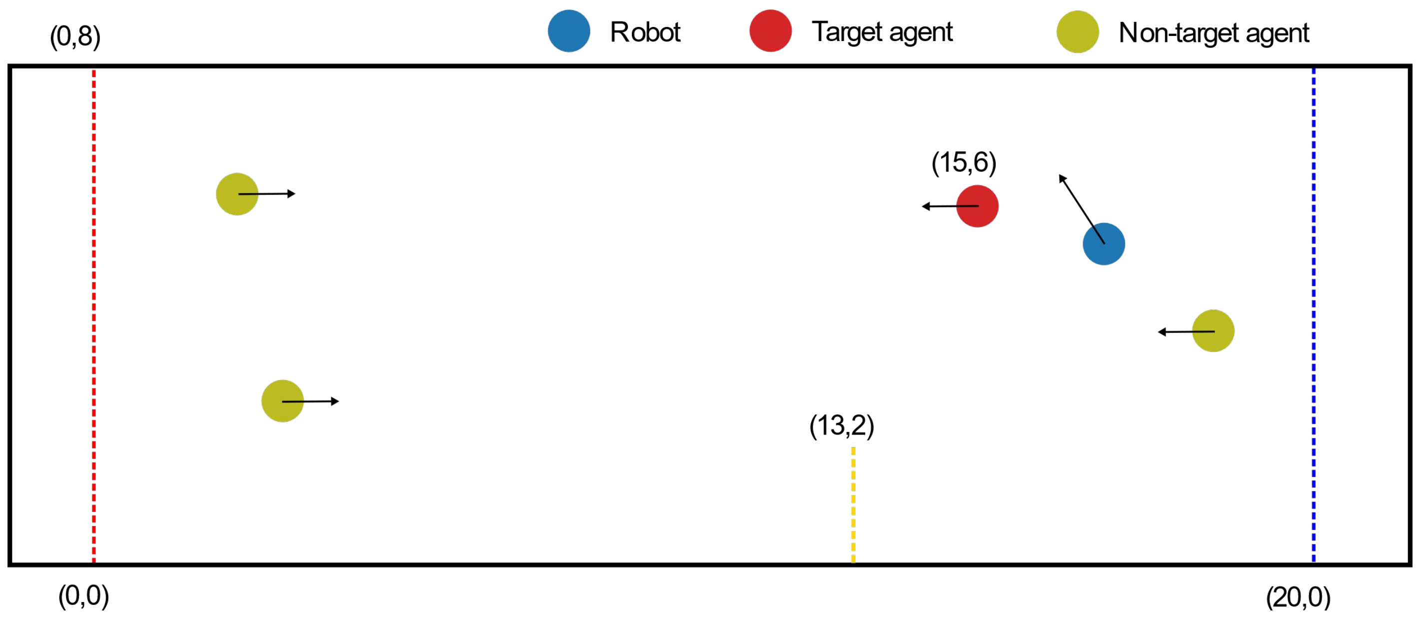
- Design of robot policies that help them move smoothly in crowded environments
- Major focus on collision avoidance
- Performance measured through disturbance to human trajectories

Influencing Human Behavior

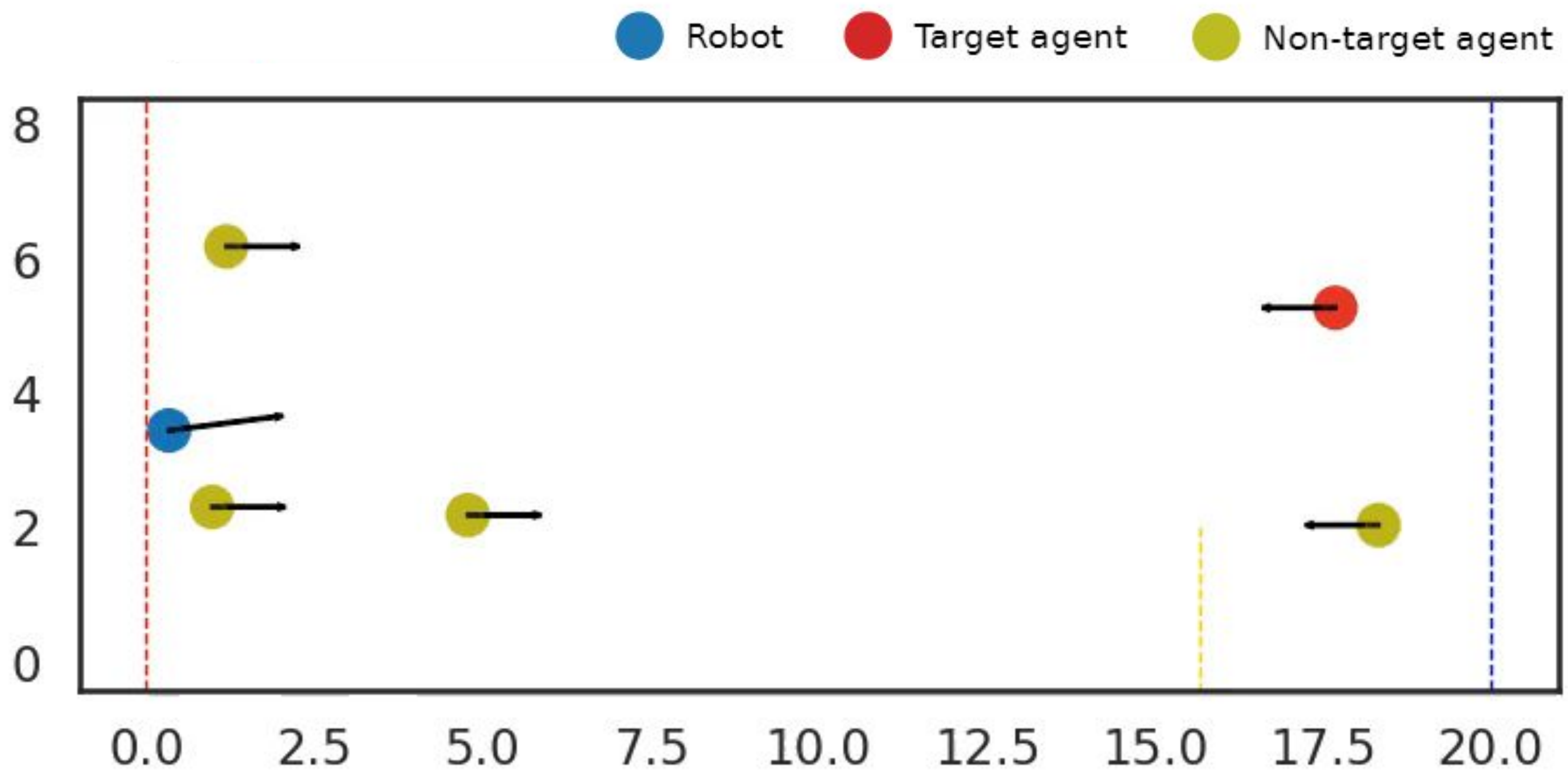
- Prior research focused on studying changes in human behavior due to robot behavior
- More recently, research has focused on designing robot behaviors that actively modify human behaviors

Goal of this study

- We consider a hallway crossing scenario
- Each agent starts at one end of the hallway and moves towards the opposite end (red and blue dashed lines)
- The robot wants the target agent to reach the virtual goal line (gold dashed line)
- The virtual goal line is always 2m ahead of the target agent



Sample initial conditions


Robot
Target agent
Non-target agent
8
6
4
2
0
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

Results

Major Assumptions

- We leverage the idea that individuals will inherently try and avoid collisions when moving in a crowded environment

- We assume that this collision avoidance behavior is modeled using Optimal Reciprocal Collision Avoidance (ORCA)

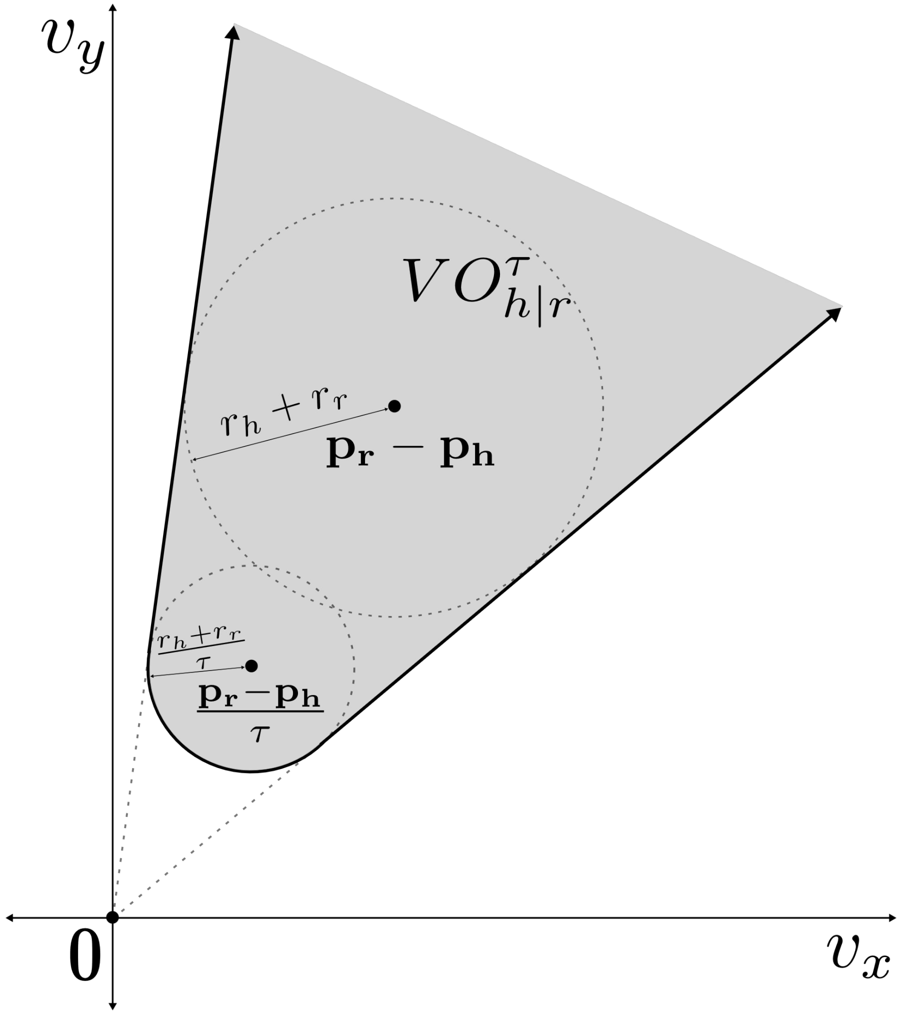
- Under this assumption, we show that we can design a policy for a robot that can implicitly nudge a target agent toward a desired direction without affecting the other agents in the environment

- We also assume that all agents in the scene use ORCA with the same policy parameters
  - This assumption is made in all research dealing with ORCA since it is the only way to guarantee collision avoidance

ORCA Preliminaries

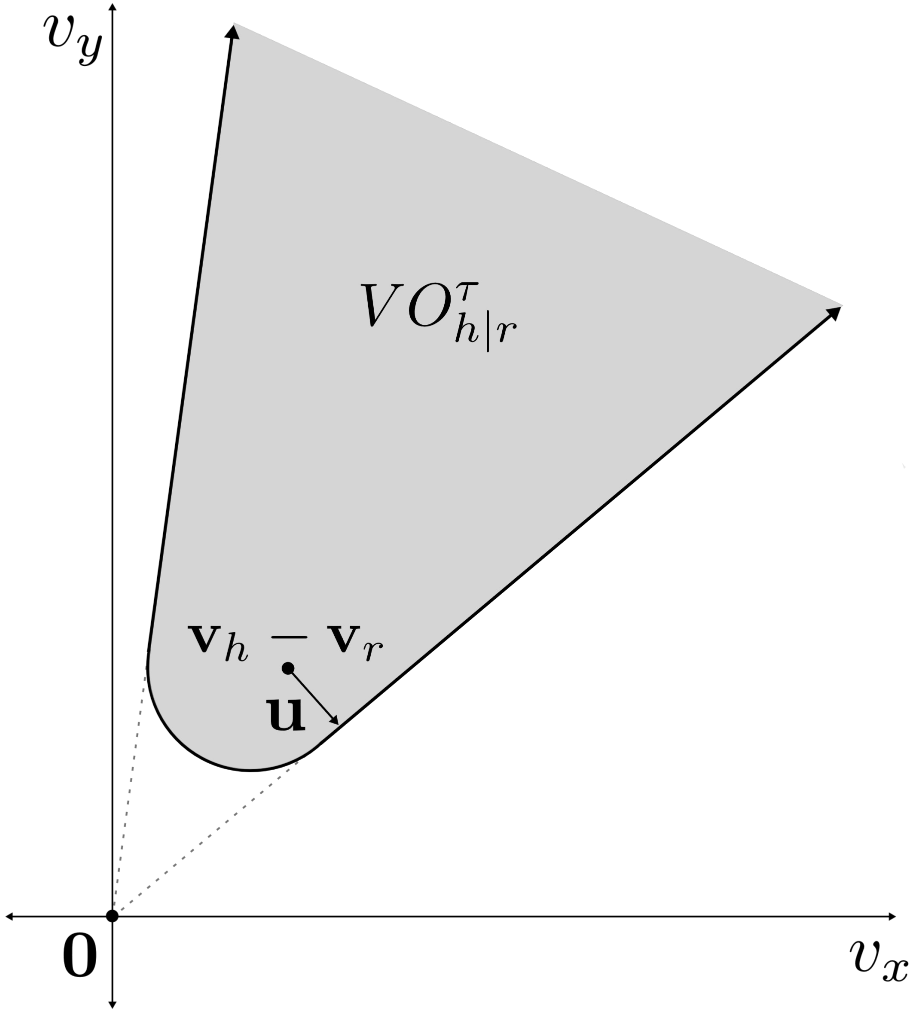
- Optimal Reciprocal Collision Avoidance (ORCA) works by using
  - Reciprocal Velocity Obstacles (RVOs) and,
  - the reciprocity assumption: Each agent shares the responsibility of avoiding collision

- For two circular agents h, r, if the relative velocity is outside the grey shaded region,
  - we can guarantee no-collisions in the next τ timesteps if the agents continue with the same velocities



ORCA Preliminaries

- If the relative velocity is within the gray region, collision can be avoided by modifying the relative velocity to be on the boundary of the velocity obstacle
- We project the relative velocity to the boundary and get $\boldsymbol{u}$
- If the relative velocity becomes $v_h - v_r + \boldsymbol{u}$, we avoid collision in the next τ timesteps
- In the original paper, the authors assumed that each agent equally shared the responsibility to avoid collisions
  - We modify this assumption by saying that each agent takes the full responsibility to avoid collision

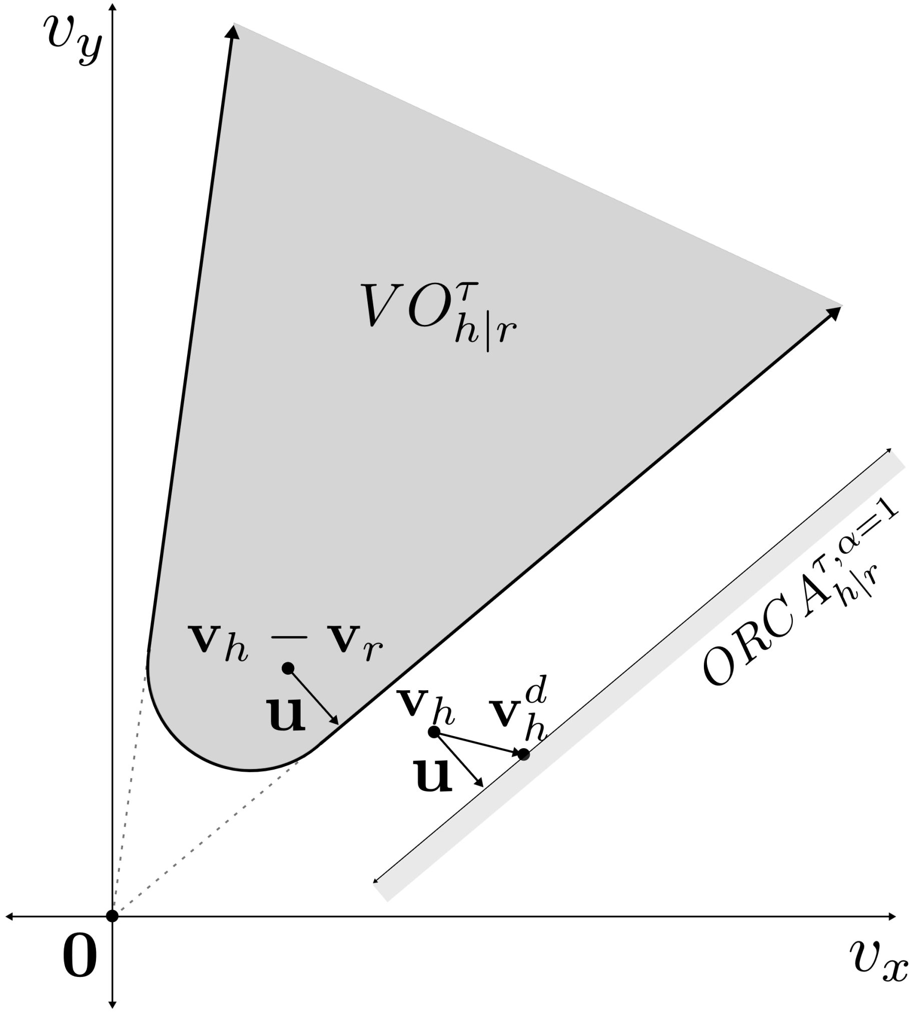


- Our key insight in this work is that
  - Agent r can set its velocity that nudges agent h's velocity towards a desired velocity

- Mathematically, this problem then becomes,

$$\min_{\mathbf{v}_r} \ ||\mathbf{v}_h^d - \mathbf{v}_h - \mathbf{u}||$$
$$\text{s.t.} \quad \mathbf{u} = \text{Proj}(\mathbf{v}_h - \mathbf{v}_r, \delta VO)$$

- We can solve this problem geometrically, giving us the I-ORCA algorithm



Algorithm 1: I-ORCA

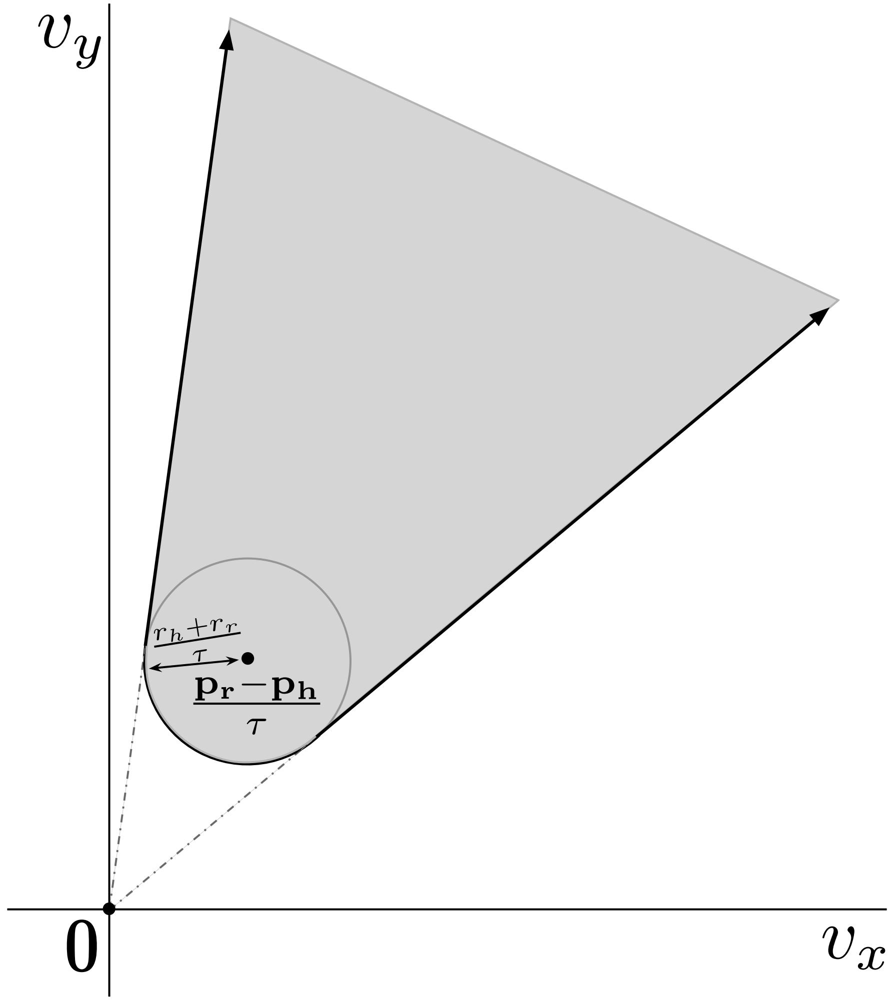
Data: $\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, r_h, r_r, \tau$
Result: $\mathbf{v}_r$
$VO \leftarrow$ GenerateVO$(\mathbf{p}_h, \mathbf{p}_r, r_h, r_r, \tau)$;
$overlap \leftarrow$ CheckOverlap$(VO, \mathbf{v}_h, v_r^{max})$;
if $overlap$ **then**
  $\mathbf{u} \leftarrow$ ComputeProjection$(VO, \mathbf{v}_h, \mathbf{v}_h^d)$;
  $\mathbf{v}_r \leftarrow \mathbf{v}_h + \mathbf{u} -$ ChoosePoint(VO);
else
  $\mathbf{u} \leftarrow$ CloserToVO$(VO, \mathbf{v}_h, v_r^{max})$;
  $\mathbf{v}_r \leftarrow \mathbf{v}_h + \mathbf{u}$;
end

Input Variable	Description
$\mathbf{v}_h$	The velocity of agent h
$\mathbf{v}_h^d$	The desired velocity for agent h
$\mathbf{p}_h$	The position of agent h
$\mathbf{p}_r$	The position of agent r
r_h	The radius of agent h
r_r	The radius of agent r
τ	The planning time horizon

Output Variable	Description
$\mathbf{v}_r$	The velocity for agent r

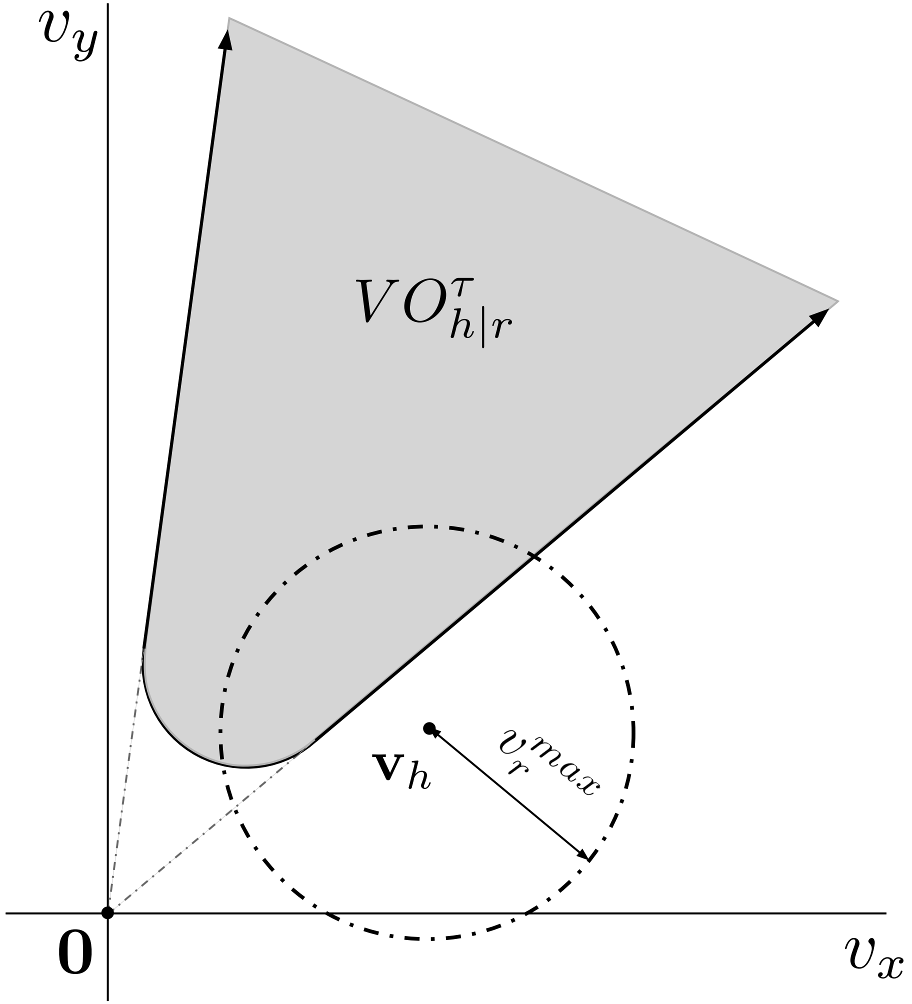
Algorithm 1: I-ORCA

Data: $\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, r_h, r_r, \tau$
Result: $\mathbf{v}_r$
$\boxed{VO \leftarrow \text{GenerateVO}(\mathbf{p}_h, \mathbf{p}_r, r_h, r_r, \tau);}$
$overlap \leftarrow \text{CheckOverlap}(VO, \mathbf{v}_h, v_r^{max});$
if $overlap$ **then**
  $\mathbf{u} \leftarrow \text{ComputeProjection}(VO, \mathbf{v}_h, \mathbf{v}_h^d);$
  $\mathbf{v}_r \leftarrow \mathbf{v}_h + \mathbf{u} - \text{ChoosePoint}(VO);$
else
  $\mathbf{u} \leftarrow \text{CloserToVO}(VO, \mathbf{v}_h, v_r^{max});$
  $\mathbf{v}_r \leftarrow \mathbf{v}_h + \mathbf{u};$
end



Algorithm 1: I-ORCA

Data: $\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, r_h, r_r, \tau$
Result: $\mathbf{v}_r$
$VO \leftarrow \text{GenerateVO}(\mathbf{p}_h, \mathbf{p}_r, r_h, r_r, \tau)$;
$\boxed{overlap \leftarrow \text{CheckOverlap}(VO, \mathbf{v}_h, v_r^{max});}$
if $overlap$ **then**
  $\mathbf{u} \leftarrow \text{ComputeProjection}(VO, \mathbf{v}_h, \mathbf{v}_h^d)$;
  $\mathbf{v}_r \leftarrow \mathbf{v}_h + \mathbf{u} - \text{ChoosePoint}(VO)$;
else
  $\mathbf{u} \leftarrow \text{CloserToVO}(VO, \mathbf{v}_h, v_r^{max})$;
  $\mathbf{v}_r \leftarrow \mathbf{v}_h + \mathbf{u}$;
end



I-ORCA

Algorithm 1: I-ORCA

Data: $\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, r_h, r_r, \tau$

Result: $\mathbf{v}_r$

$VO \leftarrow \text{GenerateVO}(\mathbf{p}_h, \mathbf{p}_r, r_h, r_r, \tau)$;

$overlap \leftarrow \text{CheckOverlap}(VO, \mathbf{v}_h, v_r^{max})$;

if $overlap$ **then**

  $\boxed{\mathbf{u} \leftarrow \text{ComputeProjection}(VO, \mathbf{v}_h, \mathbf{v}_h^d);}$

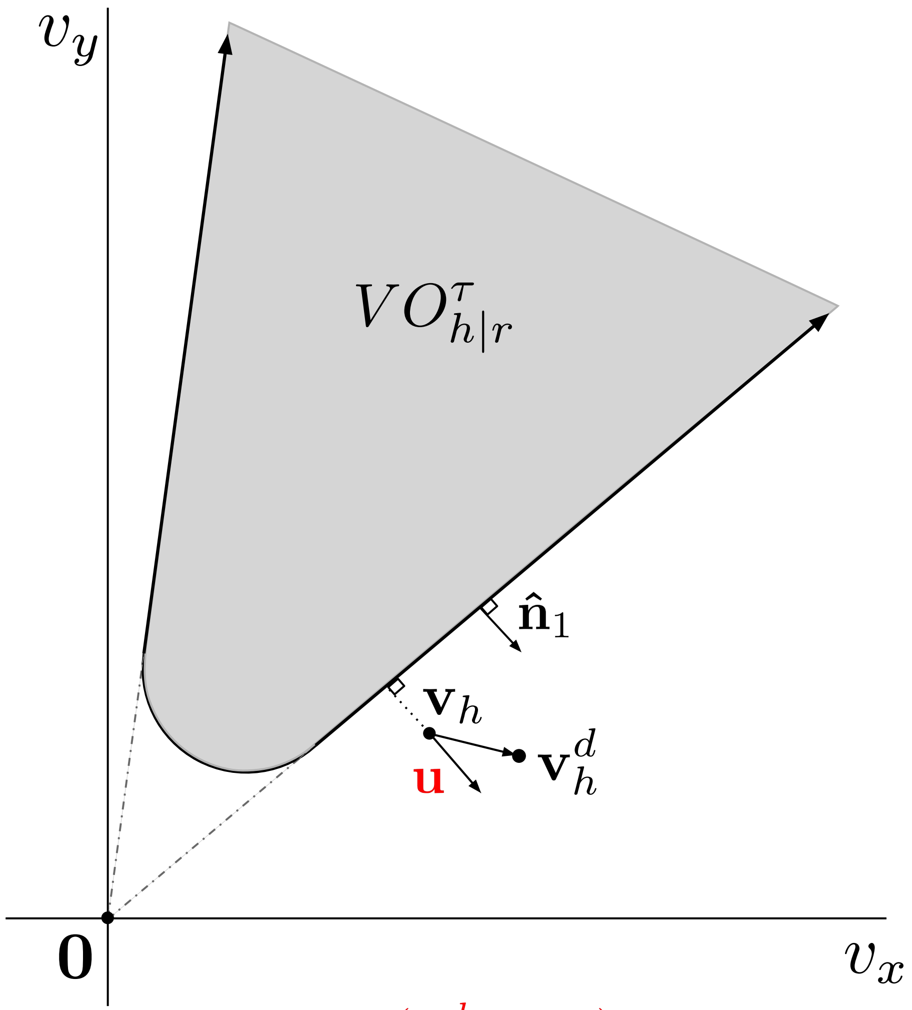
  $\mathbf{v}_r \leftarrow \mathbf{v}_h + \mathbf{u} - \text{ChoosePoint}(VO)$;

else

  $\mathbf{u} \leftarrow \text{CloserToVO}(VO, \mathbf{v}_h, v_r^{max})$;

  $\mathbf{v}_r \leftarrow \mathbf{v}_h + \mathbf{u}$;

end



$$\mathbf{u} = (\mathbf{v}_h^d - \mathbf{v}_h) \cdot \hat{\mathbf{n}}_1$$

Algorithm 1: I-ORCA

Data: $\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, r_h, r_r, \tau$

Result: $\mathbf{v}_r$

$VO \leftarrow \text{GenerateVO}(\mathbf{p}_h, \mathbf{p}_r, r_h, r_r, \tau)$;

$overlap \leftarrow \text{CheckOverlap}(VO, \mathbf{v}_h, v_r^{max})$;

if $overlap$ **then**

  $\mathbf{u} \leftarrow \text{ComputeProjection}(VO, \mathbf{v}_h, \mathbf{v}_h^d)$;

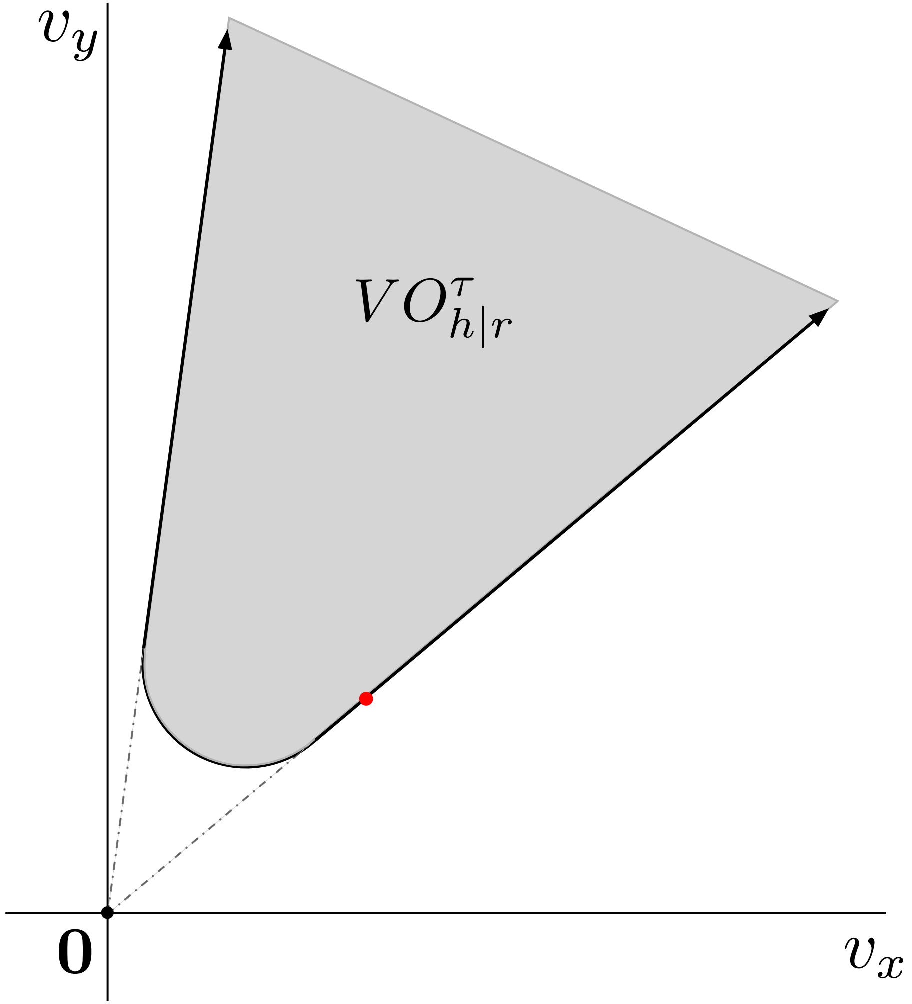
  $\mathbf{v}_r \leftarrow \mathbf{v}_h + \mathbf{u} - \boxed{\text{ChoosePoint}(VO);}$

else

  $\mathbf{u} \leftarrow \text{CloserToVO}(VO, \mathbf{v}_h, v_r^{max})$;

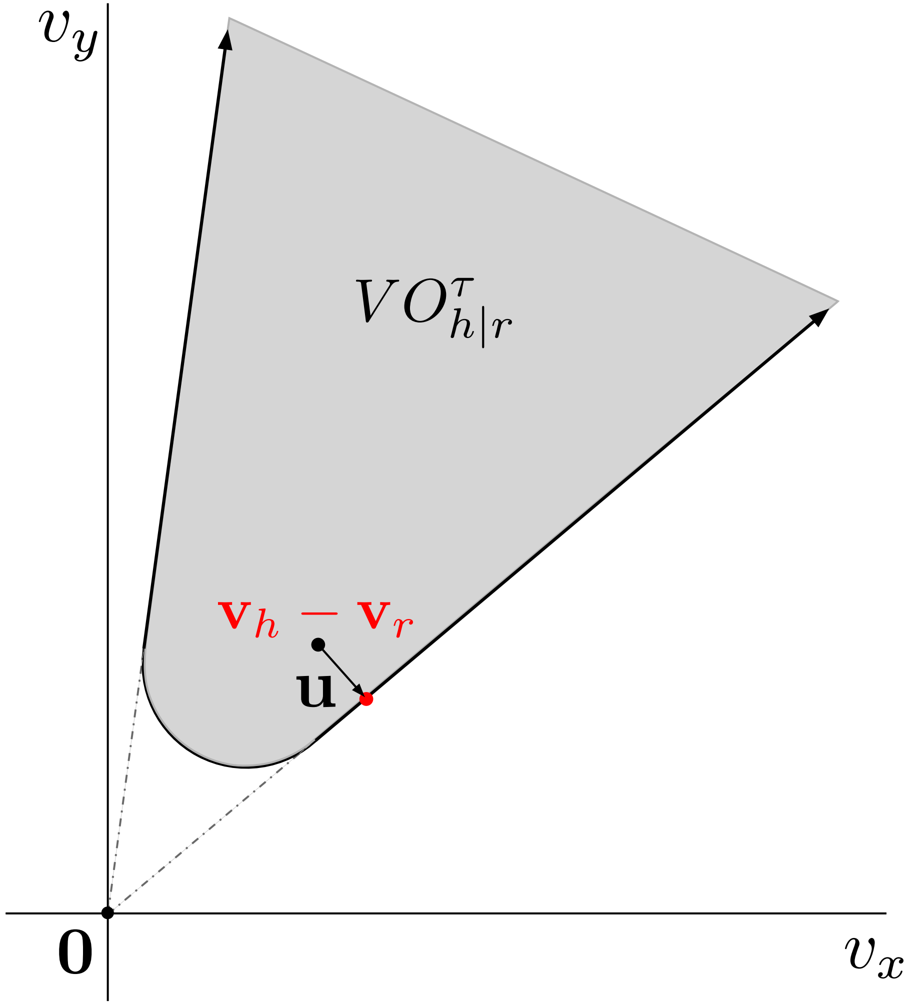
  $\mathbf{v}_r \leftarrow \mathbf{v}_h + \mathbf{u}$;

end



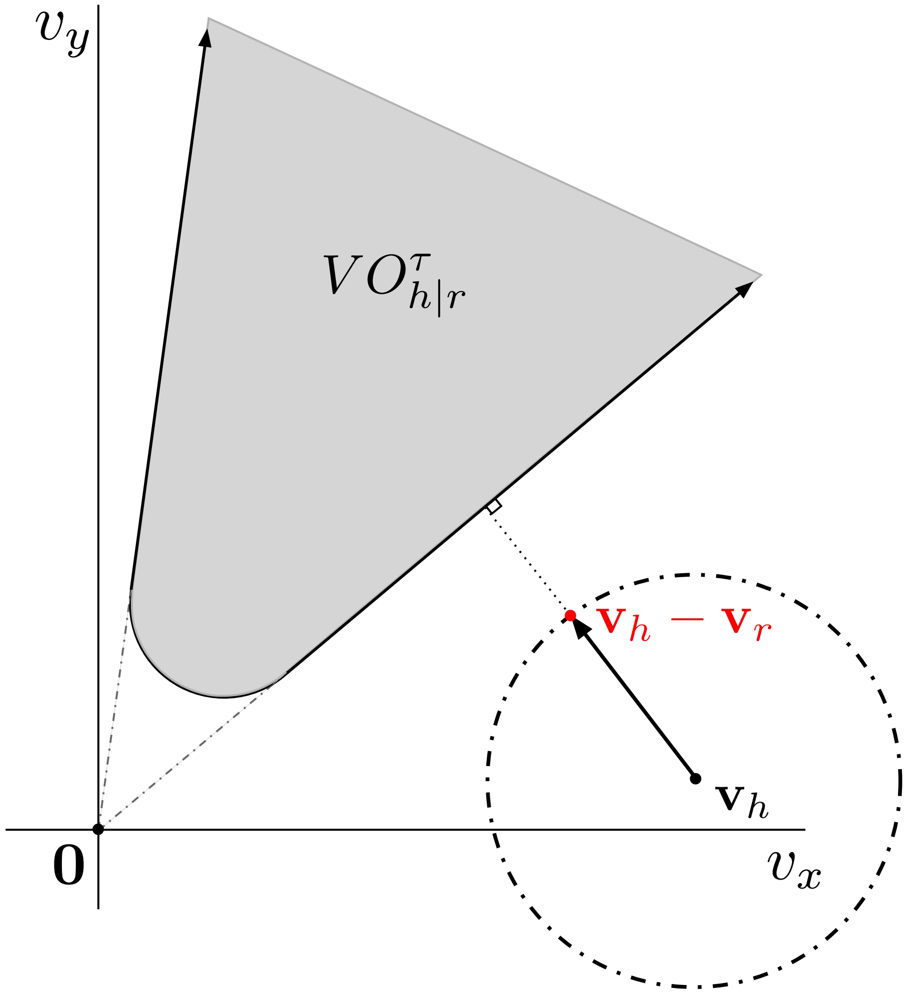
Algorithm 1: I-ORCA

Data: $\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, r_h, r_r, \tau$
Result: $\mathbf{v}_r$
$VO \leftarrow \text{GenerateVO}(\mathbf{p}_h, \mathbf{p}_r, r_h, r_r, \tau)$;
$overlap \leftarrow \text{CheckOverlap}(VO, \mathbf{v}_h, v_r^{max})$;
if *overlap* **then**
  $\mathbf{u} \leftarrow \text{ComputeProjection}(VO, \mathbf{v}_h, \mathbf{v}_h^d)$;
  $\boxed{\mathbf{v}_r \leftarrow \mathbf{v}_h + \mathbf{u} - \text{ChoosePoint}(VO)}$;
else
  $\mathbf{u} \leftarrow \text{CloserToVO}(VO, \mathbf{v}_h, v_r^{max})$;
  $\mathbf{v}_r \leftarrow \mathbf{v}_h + \mathbf{u}$;
end



Algorithm 1: I-ORCA

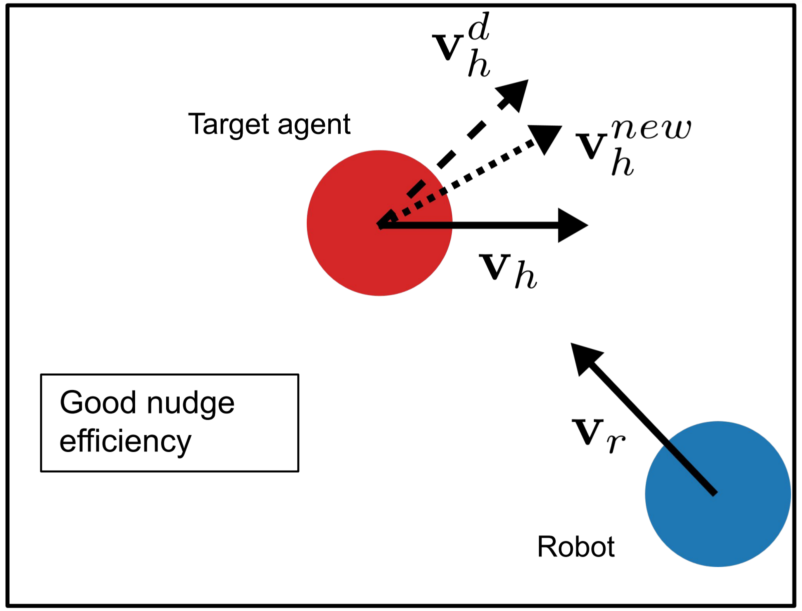
Data: $\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, r_h, r_r, \tau$
Result: $\mathbf{v}_r$
$VO \leftarrow \text{GenerateVO}(\mathbf{p}_h, \mathbf{p}_r, r_h, r_r, \tau)$;
$overlap \leftarrow \text{CheckOverlap}(VO, \mathbf{v}_h, v_r^{max})$;
if $overlap$ **then**
    $\mathbf{u} \leftarrow \text{ComputeProjection}(VO, \mathbf{v}_h, \mathbf{v}_h^d)$;
    $\mathbf{v}_r \leftarrow \mathbf{v}_h + \mathbf{u} - \text{ChoosePoint}(VO)$;
else
    $\boxed{\mathbf{u} \leftarrow \text{CloserToVO}(VO, \mathbf{v}_h, v_r^{max});}$
    $\mathbf{v}_r \leftarrow \mathbf{v}_h + \mathbf{u}$;
end

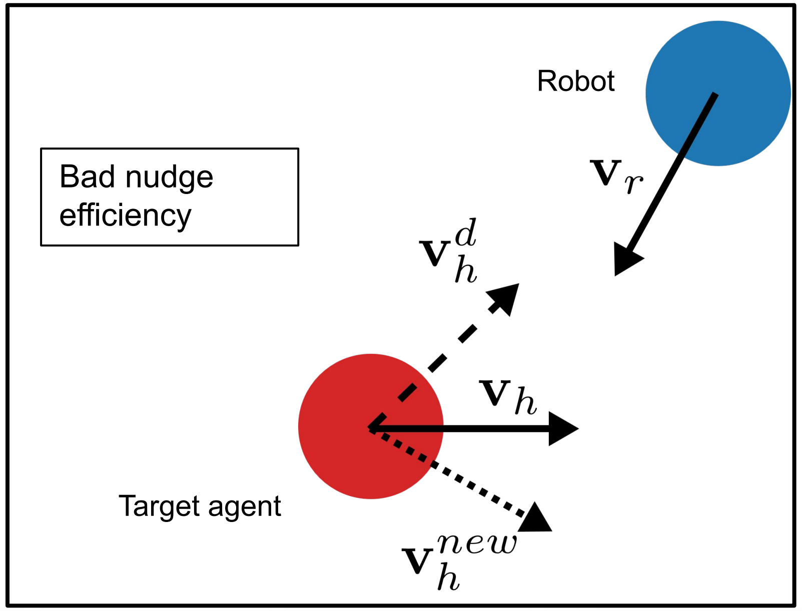


Nudge Efficiency

- Not all positions around the target agent are equally good at nudging it towards the desired direction

- Nudge efficiency is defined as the similarity between the target agent's new velocity after nudging to the desired velocity





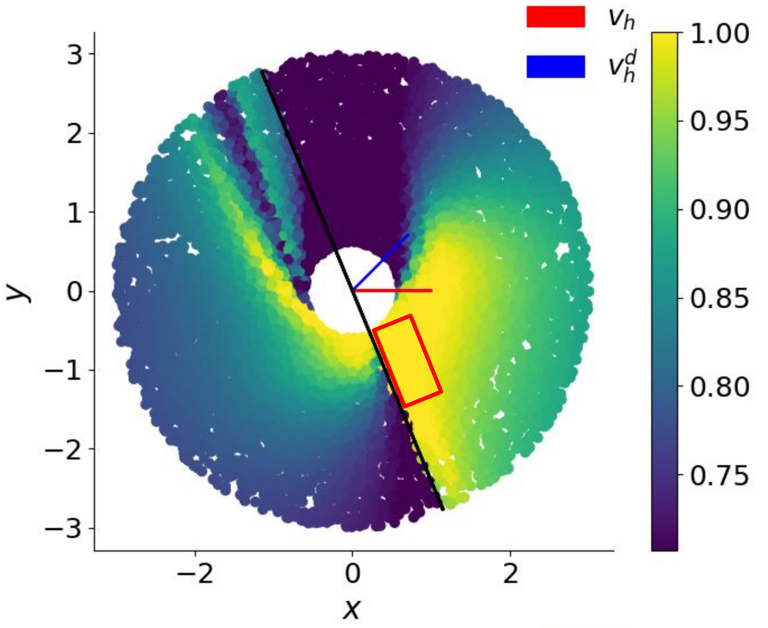
Nudge Efficiency

- The nudge efficiency was computed by
  - Sampling the robot’s position around the target agent
  - Setting the robot’s velocity using I-ORCA
  - Getting the target agent’s velocity at the next time step using its motion model
  - Computing the cosine similarity between the desired velocity and the new velocity

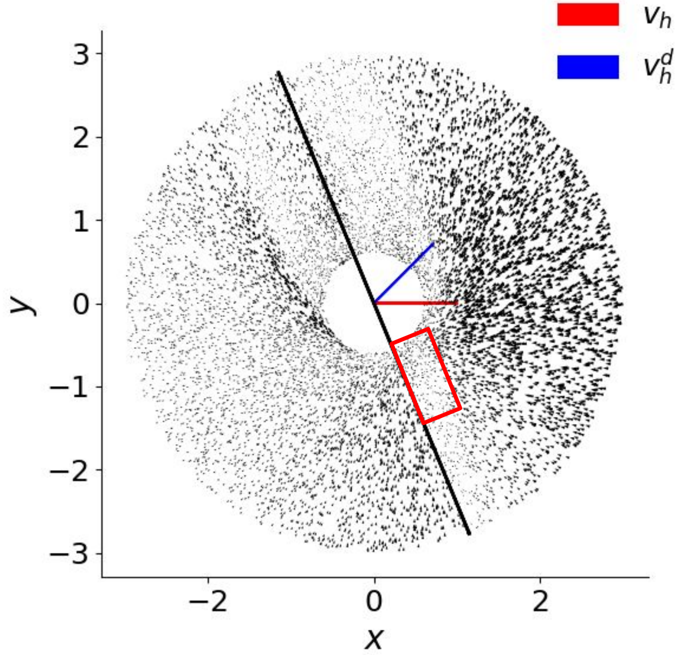
- The illustration to the right shows the case when the target agent uses ORCA

- We see a similar plot when the target agent uses the social forces model

Nudge efficiency



Relative velocities



Smooth Efficient Nudging Policy

Algorithm 2: Smooth Efficient Nudge Policy

Data: $\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, r_h, r_r, v_r^{max}, \tau$
Result: $\mathbf{v}_r$
$\mathbf{v}_r^I \leftarrow$ I-ORCA(**Data**);
$\mathbf{p}_c \leftarrow \text{ChoosePoint}(\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h)$;
if *CheckConditions*$(\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, \mathbf{p}_c)$ **then**
  $\mathbf{v}_r^{pref} \leftarrow \mathbf{v}_r^I$;
  $\mathbf{v}_r \leftarrow \text{ORCA}^n(\mathbf{v}_r^{pref}, \text{other agents' data})$;
else
  $\mathbf{v}_r^{pref} \leftarrow \text{VelocityTowards}(\mathbf{p}_c)$;
  $\mathbf{v}_r^O \leftarrow \text{ORCA}^{n+1}(\mathbf{v}_r^{pref}, \text{other agents' data})$;
  $w_I \leftarrow \text{ComputeWeight}(\mathbf{p}_c, \mathbf{p}_r)$;
  $\mathbf{v}_r \leftarrow w_I \mathbf{v}_r^I + (1 - w_I)\mathbf{v}_r^O$;
end

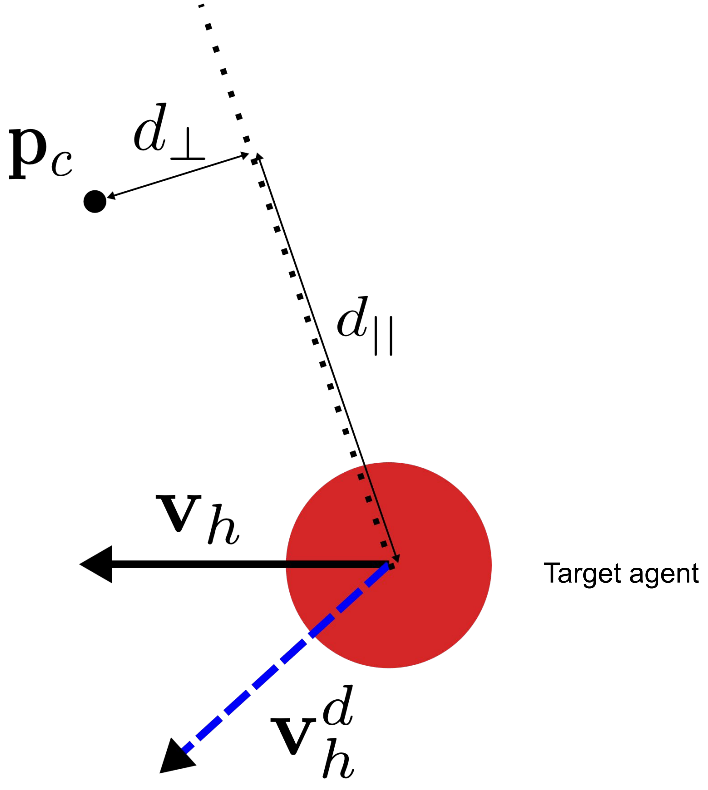
Input Variable	Description
$\mathbf{v}_h$	The velocity of agent h
$\mathbf{v}_h^d$	The desired velocity for agent h
$\mathbf{p}_h$	The position of agent h
$\mathbf{p}_r$	The position of agent r
r_h	The radius of agent h
r_r	The radius of agent r
τ	The planning time horizon

Output Variable	Description
$\mathbf{v}_r$	The velocity for agent r

Smooth Efficient Nudging Policy

Algorithm 2: Smooth Efficient Nudge Policy

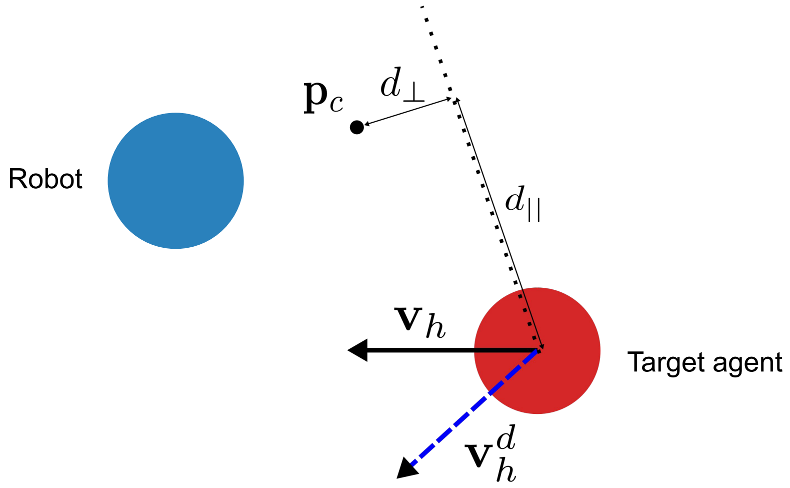
Data: $\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, r_h, r_r, v_r^{max}, \tau$
Result: $\mathbf{v}_r$
$\mathbf{v}_r^I \leftarrow$ I-ORCA(**Data**);
$\boxed{\mathbf{p}_c \leftarrow \text{ChoosePoint}(\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h);}$
if *CheckConditions*($\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, \mathbf{p}_c$) **then**
  $\mathbf{v}_r^{pref} \leftarrow \mathbf{v}_r^I$;
  $\mathbf{v}_r \leftarrow \text{ORCA}^n(\mathbf{v}_r^{pref}$, other agents' data);
else
  $\mathbf{v}_r^{pref} \leftarrow \text{VelocityTowards}(\mathbf{p}_c)$;
  $\mathbf{v}_r^O \leftarrow \text{ORCA}^{n+1}(\mathbf{v}_r^{pref}$, other agents' data);
  $w_I \leftarrow \text{ComputeWeight}(\mathbf{p}_c, \mathbf{p}_r)$;
  $\mathbf{v}_r \leftarrow w_I \mathbf{v}_r^I + (1 - w_I)\mathbf{v}_r^O$;
end



Smooth Efficient Nudging Policy

Algorithm 2: Smooth Efficient Nudge Policy

Data: $\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, r_h, r_r, v_r^{max}, \tau$
Result: $\mathbf{v}_r$
$\mathbf{v}_r^I \leftarrow$ I-ORCA(**Data**);
$\mathbf{p}_c \leftarrow$ ChoosePoint($\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h$);
if $\boxed{\mathit{CheckConditions}(\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, \mathbf{p}_c)}$ **then**
  $\mathbf{v}_r^{pref} \leftarrow \mathbf{v}_r^I$;
  $\mathbf{v}_r \leftarrow \text{ORCA}^n(\mathbf{v}_r^{pref}, \text{other agents' data})$;
else
  $\mathbf{v}_r^{pref} \leftarrow$ VelocityTowards($\mathbf{p}_c$);
  $\mathbf{v}_r^O \leftarrow \text{ORCA}^{n+1}(\mathbf{v}_r^{pref}, \text{other agents' data})$;
  $w_I \leftarrow$ ComputeWeight($\mathbf{p}_c, \mathbf{p}_r$);
  $\mathbf{v}_r \leftarrow w_I \mathbf{v}_r^I + (1 - w_I)\mathbf{v}_r^O$;
end



Checks the following conditions:

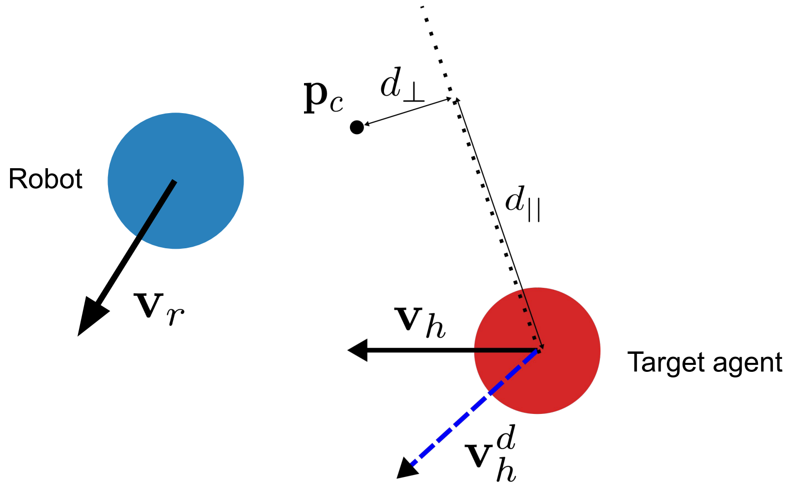
1. The ego agent should be close to the point given by ChoosePoint
2. The ego agent must be on the same side of the boundary as the chosen point
3. The ego agent's y-coordinate should be more than that of the target agent

Note: The third condition is context dependent. It works in our case since we want to nudge the target agent toward lower y-coordinates

Smooth Efficient Nudging Policy

Algorithm 2: Smooth Efficient Nudge Policy

Data: $\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, r_h, r_r, v_r^{max}, \tau$
Result: $\mathbf{v}_r$
$\mathbf{v}_r^I \leftarrow$ I-ORCA(**Data**);
$\mathbf{p}_c \leftarrow$ ChoosePoint$(\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h)$;
if *CheckConditions*$(\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, \mathbf{p}_c)$ **then**
  $\mathbf{v}_r^{pref} \leftarrow \mathbf{v}_r^I$;
  $\boxed{\mathbf{v}_r \leftarrow \mathrm{ORCA}^n(\mathbf{v}_r^{pref}, \text{other agents' data});}$
else
  $\mathbf{v}_r^{pref} \leftarrow$ VelocityTowards$(\mathbf{p}_c)$;
  $\mathbf{v}_r^O \leftarrow \mathrm{ORCA}^{n+1}(\mathbf{v}_r^{pref}, \text{other agents' data})$;
  $w_I \leftarrow$ ComputeWeight$(\mathbf{p}_c, \mathbf{p}_r)$;
  $\mathbf{v}_r \leftarrow w_I \mathbf{v}_r^I + (1 - w_I)\mathbf{v}_r^O$;
end



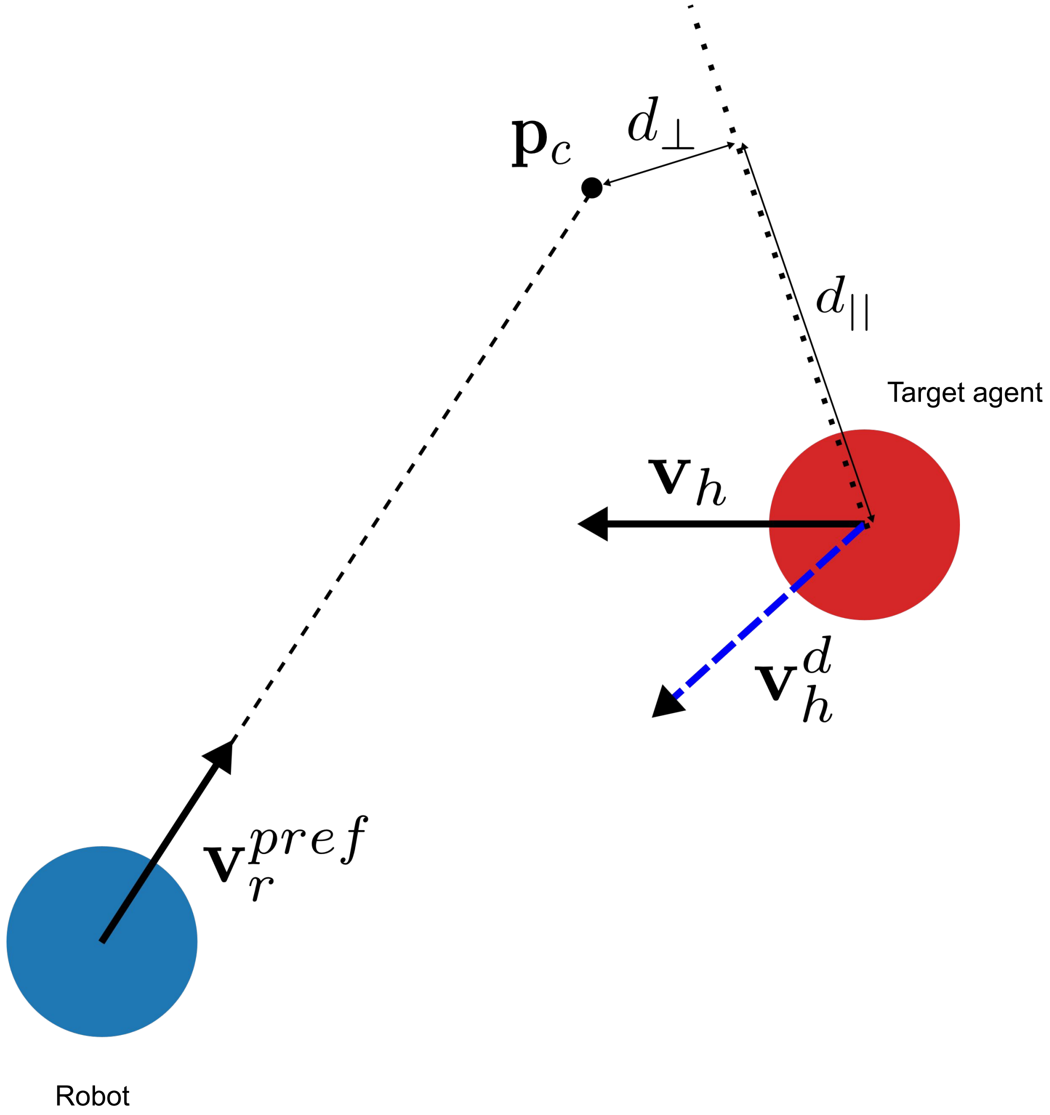
- Here, we nudge the target human
- Therefore, we select the robot's velocity using $ORCA^n$, which only considers the non-target agents in planning

Caveat:
- This can lead to collisions between the robot and the target agent
- We saw collisions in <5% of the simulations we ran
- Some policy parameters can be tuned to better avoid collisions

Smooth Efficient Nudging Policy

Algorithm 2: Smooth Efficient Nudge Policy

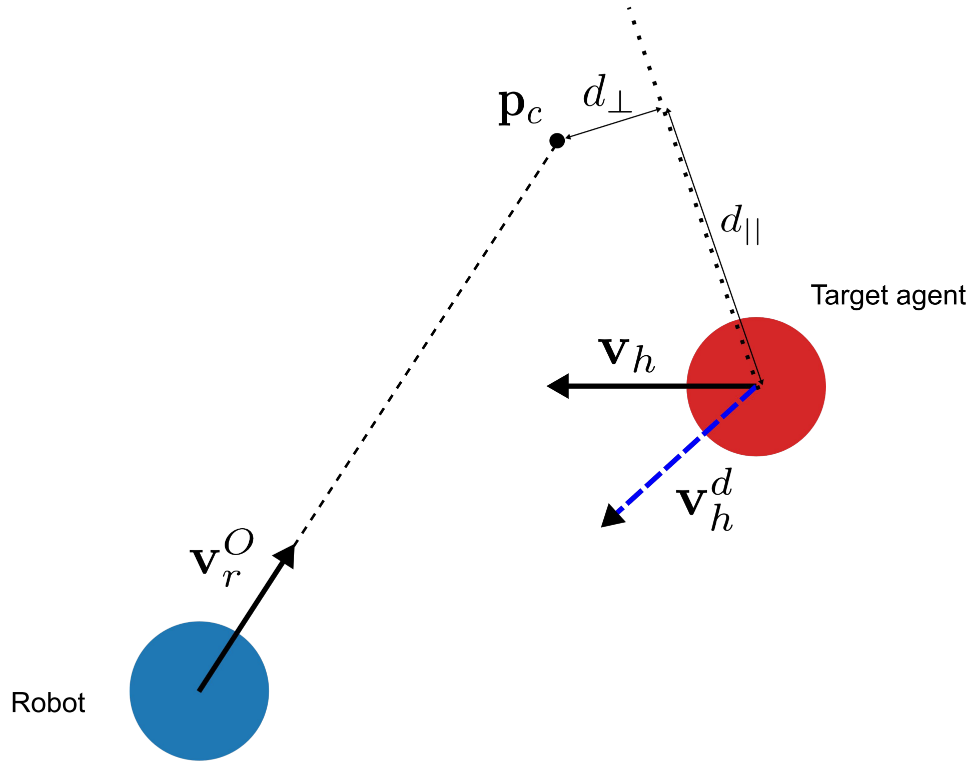
Data: $\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, r_h, r_r, v_r^{max}, \tau$
Result: $\mathbf{v}_r$
$\mathbf{v}_r^I \leftarrow$ I-ORCA(**Data**);
$\mathbf{p}_c \leftarrow$ ChoosePoint$(\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h)$;
if *CheckConditions*$(\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, \mathbf{p}_c)$ **then**
    $\mathbf{v}_r^{pref} \leftarrow \mathbf{v}_r^I$;
    $\mathbf{v}_r \leftarrow \mathrm{ORCA}^n(\mathbf{v}_r^{pref}$, other agents' data);
else
    $\boxed{\mathbf{v}_r^{pref} \leftarrow \mathrm{VelocityTowards}(\mathbf{p}_c);}$
    $\mathbf{v}_r^O \leftarrow \mathrm{ORCA}^{n+1}(\mathbf{v}_r^{pref}$, other agents' data);
    $w_I \leftarrow \mathrm{ComputeWeight}(\mathbf{p}_c, \mathbf{p}_r)$;
    $\mathbf{v}_r \leftarrow w_I \mathbf{v}_r^I + (1 - w_I)\mathbf{v}_r^O$;
end



Smooth Efficient Nudging Policy

Algorithm 2: Smooth Efficient Nudge Policy

Data: $\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, r_h, r_r, v_r^{max}, \tau$
Result: $\mathbf{v}_r$
$\mathbf{v}_r^I \leftarrow$ I-ORCA(**Data**);
$\mathbf{p}_c \leftarrow$ ChoosePoint($\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h$);
if *CheckConditions*($\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, \mathbf{p}_c$) **then**
  $\mathbf{v}_r^{pref} \leftarrow \mathbf{v}_r^I$;
  $\mathbf{v}_r \leftarrow$ ORCA^n($\mathbf{v}_r^{pref}$, other agents' data);
else
  $\mathbf{v}_r^{pref} \leftarrow$ VelocityTowards($\mathbf{p}_c$);
  $\boxed{\mathbf{v}_r^O \leftarrow \text{ORCA}^{n+1}(\mathbf{v}_r^{pref}, \text{other agents' data});}$
  $w_I \leftarrow$ ComputeWeight($\mathbf{p}_c, \mathbf{p}_r$);
  $\mathbf{v}_r \leftarrow w_I \mathbf{v}_r^I + (1 - w_I)\mathbf{v}_r^O$;
end

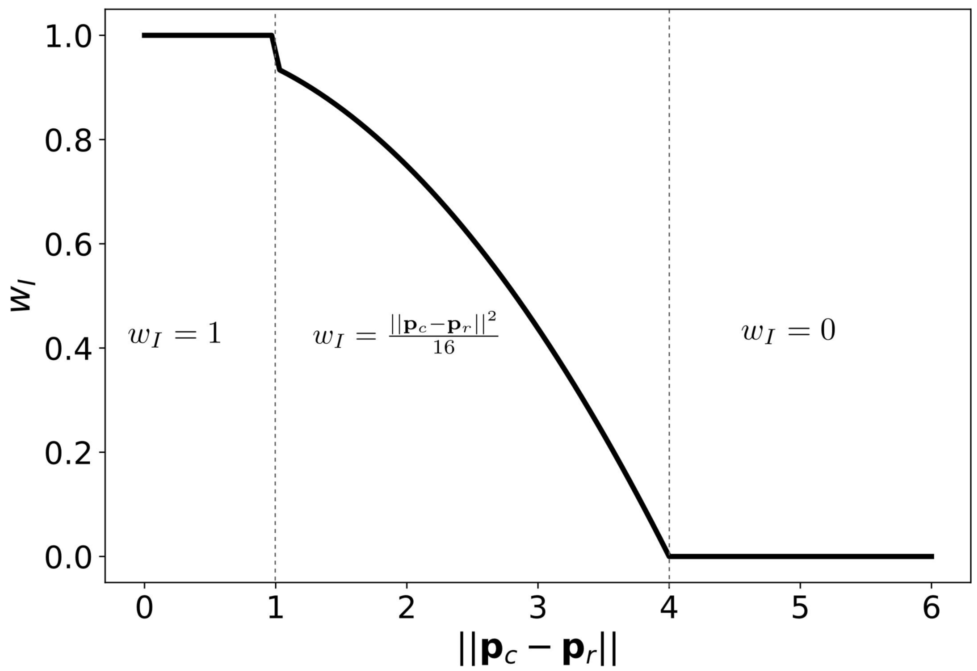


- Here, we do not nudge the target human until we reach the efficient nudging region
- Therefore, we select the robot's velocity using $ORCA^{n+1}$, which considers all agents in the scene during planning

Smooth Efficient Nudging Policy

Algorithm 2: Smooth Efficient Nudge Policy

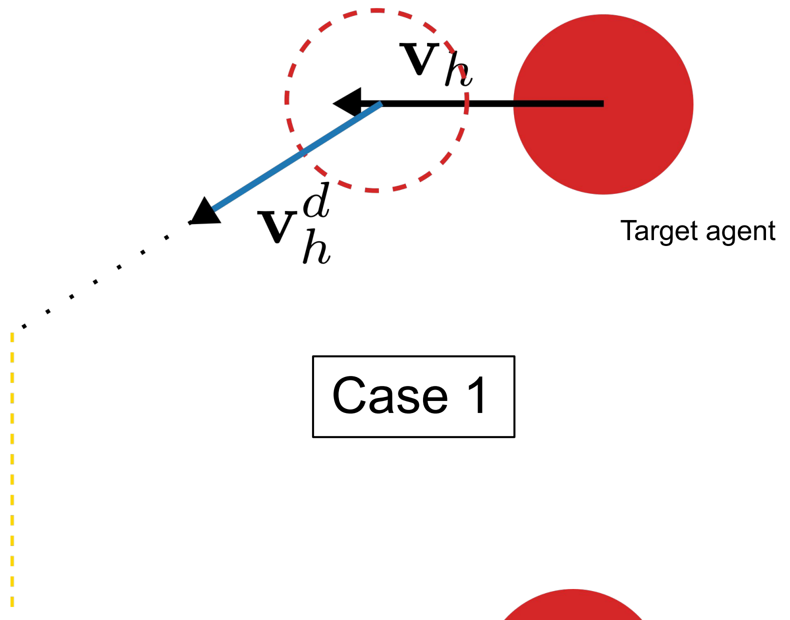
Data: $\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, r_h, r_r, v_r^{max}, \tau$
Result: $\mathbf{v}_r$
$\mathbf{v}_r^I \leftarrow$ I-ORCA(**Data**);
$\mathbf{p}_c \leftarrow$ ChoosePoint$(\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h)$;
if *CheckConditions*$(\mathbf{v}_h, \mathbf{v}_h^d, \mathbf{p}_h, \mathbf{p}_r, \mathbf{p}_c)$ **then**
  $\mathbf{v}_r^{pref} \leftarrow \mathbf{v}_r^I$;
  $\mathbf{v}_r \leftarrow \text{ORCA}^n(\mathbf{v}_r^{pref}, \text{other agents' data})$;
else
  $\mathbf{v}_r^{pref} \leftarrow \text{VelocityTowards}(\mathbf{p}_c)$;
  $\mathbf{v}_r^O \leftarrow \text{ORCA}^{n+1}(\mathbf{v}_r^{pref}, \text{other agents' data})$;
  $w_I \leftarrow \text{ComputeWeight}(\mathbf{p}_c, \mathbf{p}_r)$;
  $\mathbf{v}_r \leftarrow w_I \mathbf{v}_r^I + (1 - w_I)\mathbf{v}_r^O$;
end

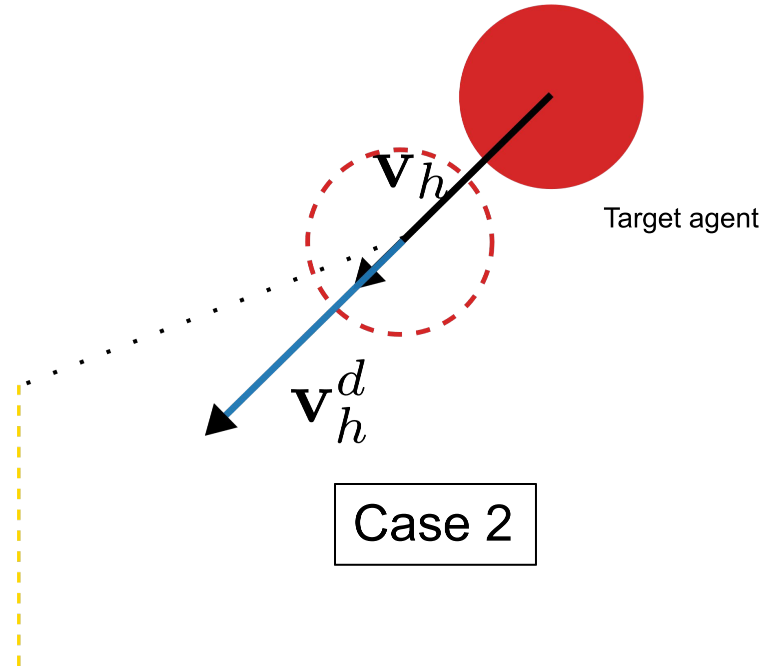


Any non-increasing function of the distance between the ego agent and the chosen point should work here

Setting the desired velocity v_h^d

- The desired velocity is set using the next position of the target agent, the virtual goal line, and the current velocity of the target agent
- First, the velocity with max speed is computed from the next anticipated position to the top of the virtual goal line
- If the y-component of this velocity is less than that of the target agent,
  - Set this velocity as the desired velocity
- Else,
  - Set the current velocity of the target agent as the desired velocity





Results - Nudge success

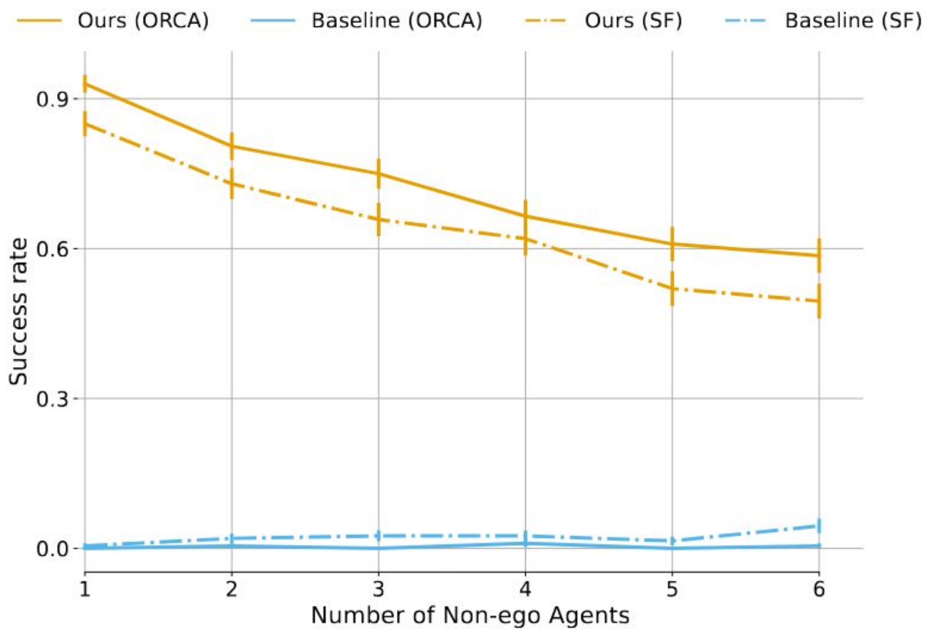


Fig. 6: Ratio of times the target agent reached the virtual goal line (SF - Social Force)

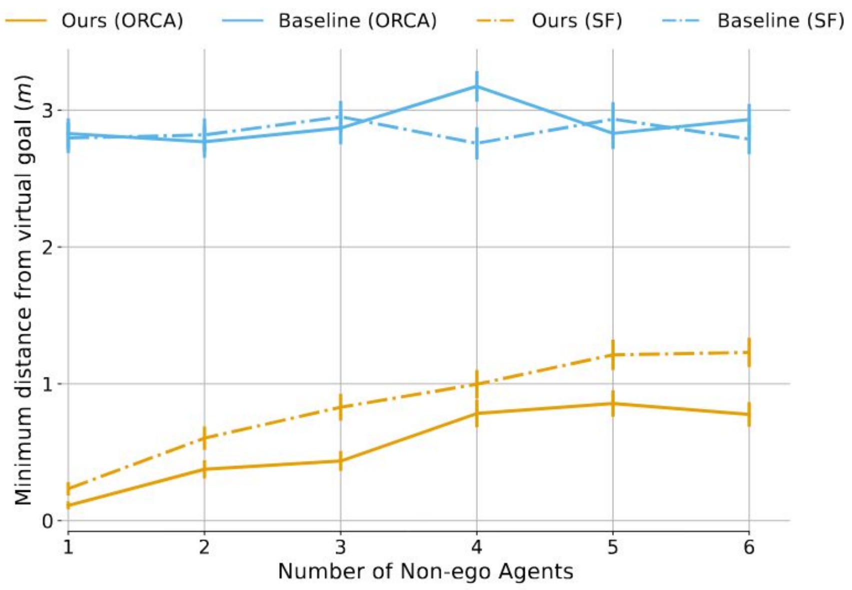
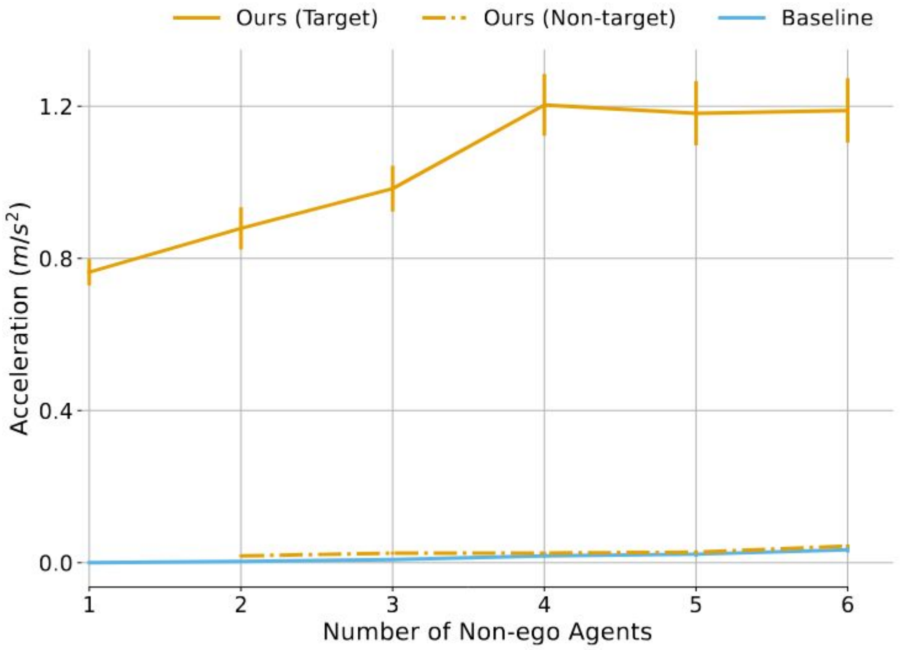
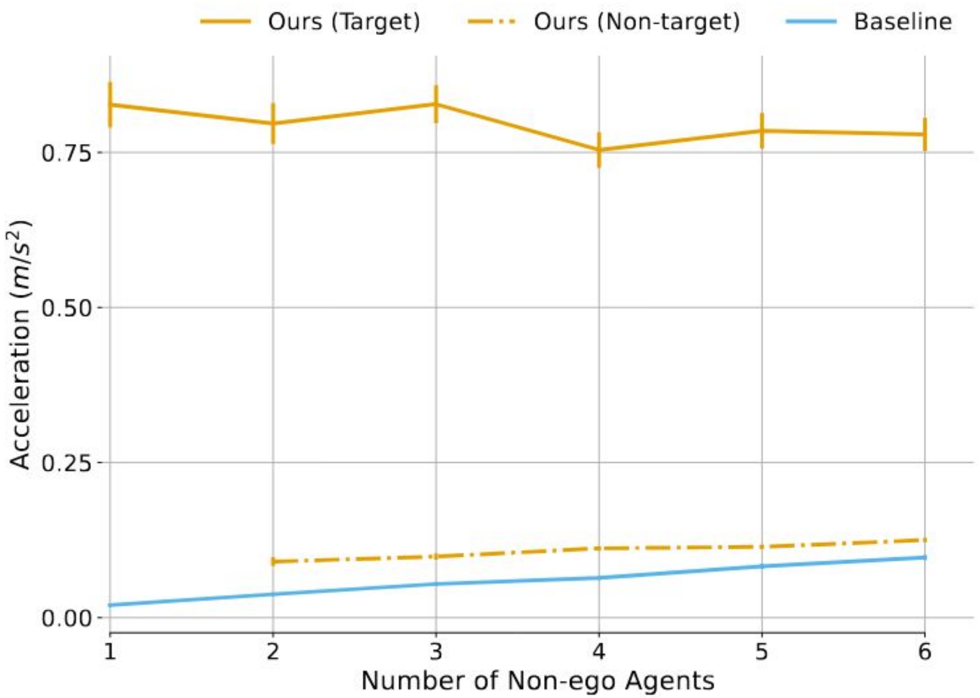


Fig. 7: The minimum distance between the target agent's y-coordinate and the top of the virtual goal line (SF - Social Force)

Results – Disturbance to non-target agents



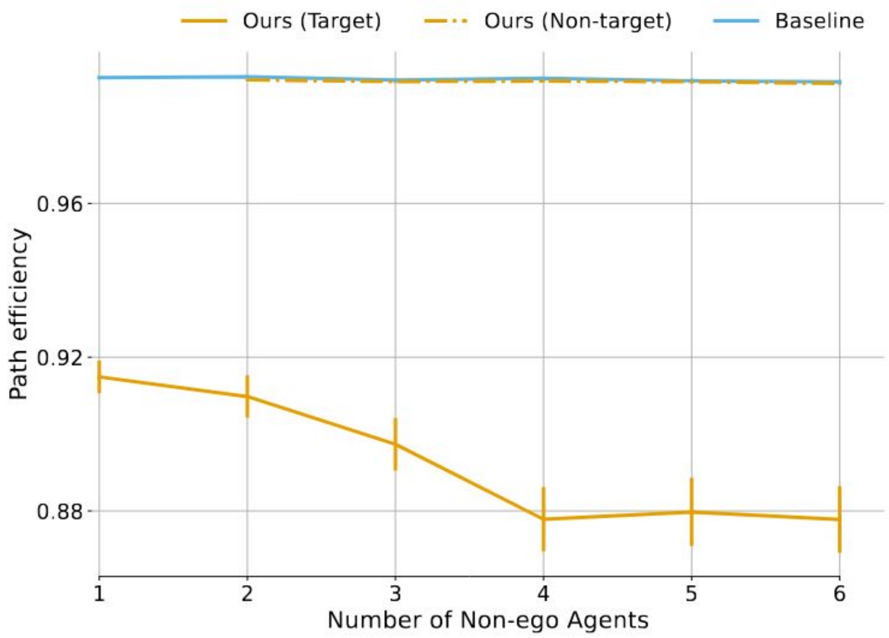
(a) Non-ego Agents use ORCA



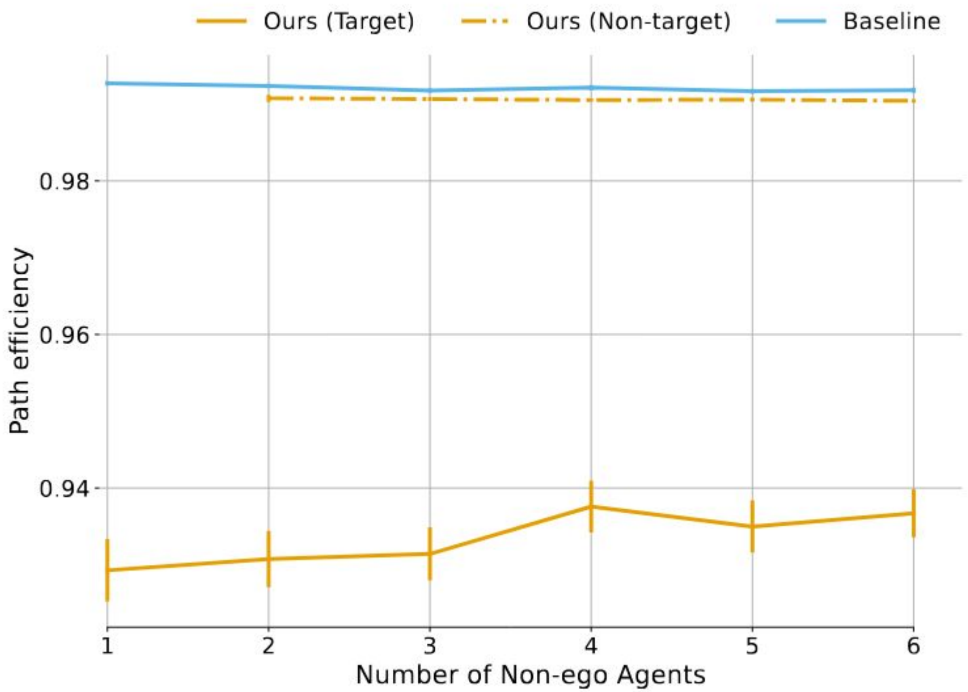
(b) Non-ego Agents use Social Forces

Fig. 8: The average acceleration of the non-ego agents over their trajectories. (a) is the case when all non-ego agents use ORCA and (b) is the case when all non-ego agents use the social forces model

Results – Disturbance to non-target agents



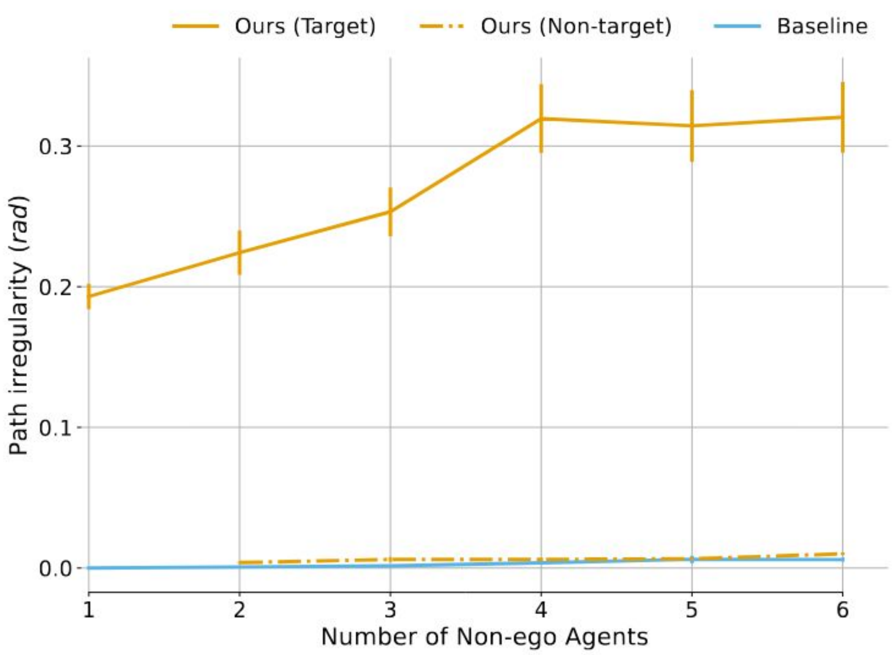
(a) Non-ego agents use ORCA



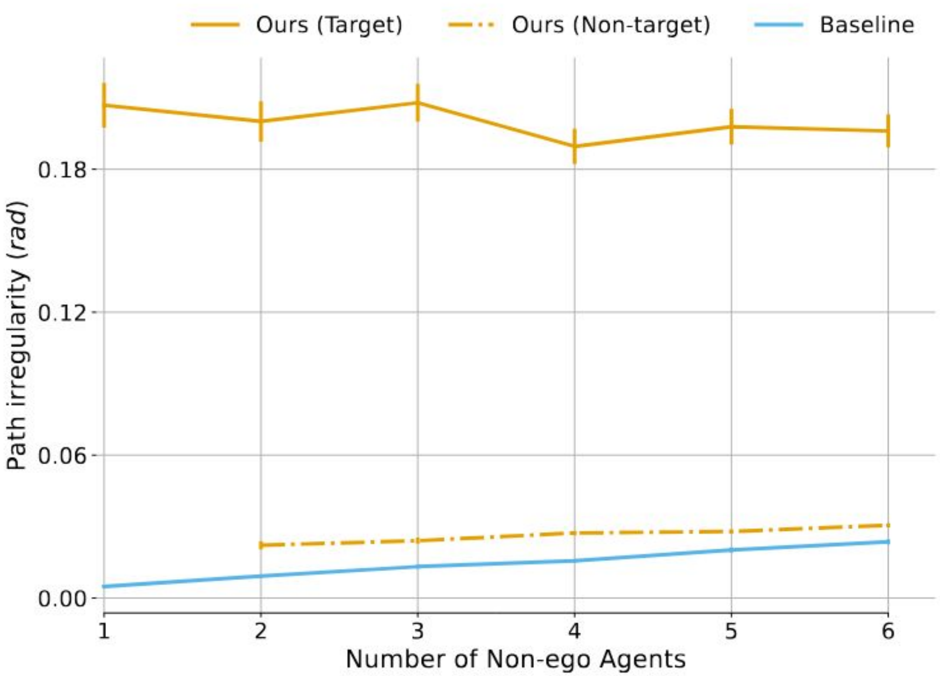
(b) Non-ego agents use Social Forces

Fig. 9: The path efficiency of the non-ego agents. (a) is the case when all non-ego agents use ORCA and (b) is the case when all non-ego agents use the social forces model

Results – Disturbance to non-target agents



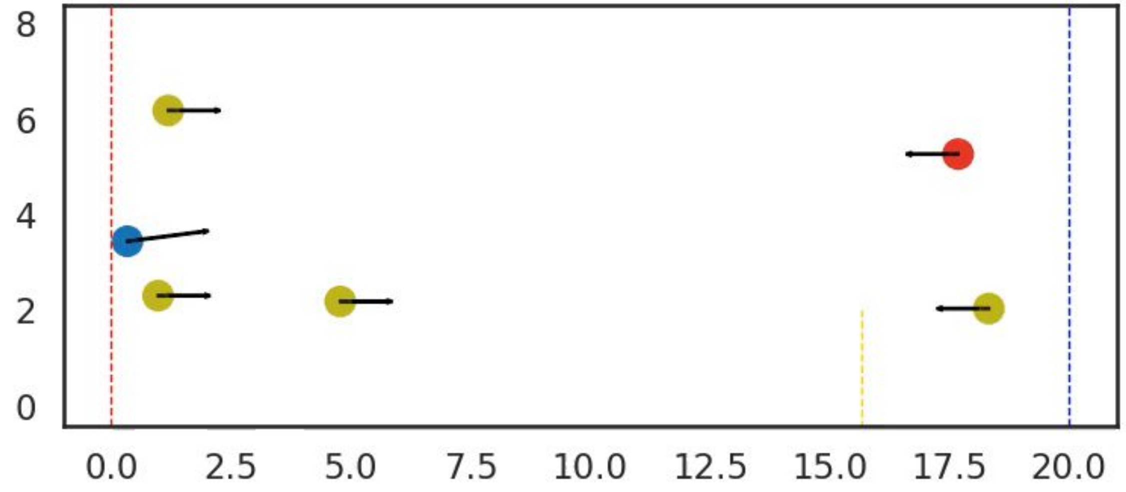
(a) Non-ego agents use ORCA



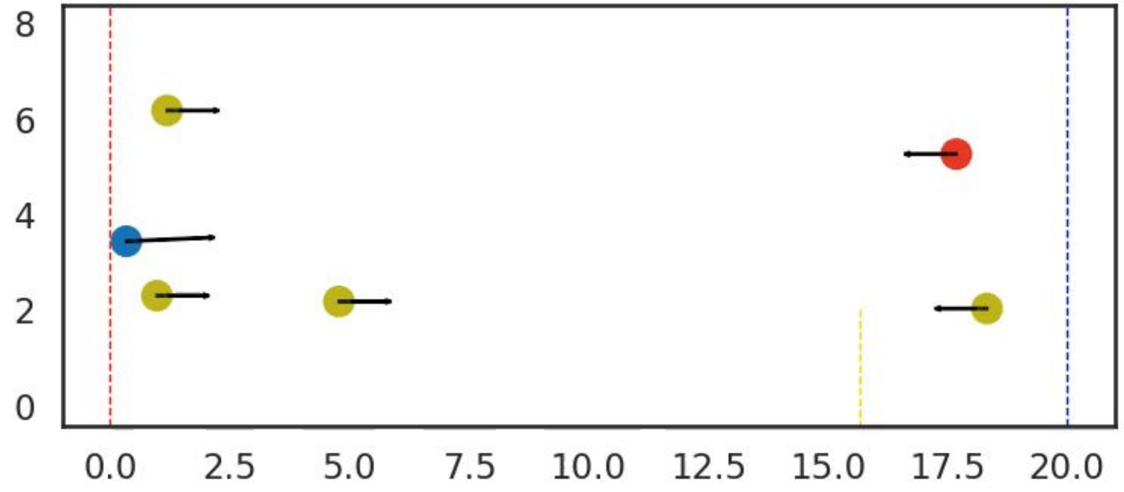
(b) Non-ego agents use Social Forces

Fig. 10: The path irregularity of the non-ego agents. (a) is the case when all non-ego agents use ORCA and (b) is the case when all non-ego agents use the social forces model

Illustrative example



Our Policy



Baseline

Summary

- We proposed the I-ORCA algorithm that inverts ORCA to generate velocities for an agent to optimally nudge another agent in the desired direction
- We computed the nudge efficiency metric, which showed that “leading” the agent toward the desired direction is the best way to nudge it
- We proposed the smooth efficient nudge policy that utilizes this result to smoothly nudge the target agent toward the desired objective
- Our results indicate that an agent using our policy was able to implicitly nudge the target agent while not disturbing the other non-target agents in the scene
  - These results were generalizable across the ORCA and Social Forces motion models

Limitations and Future Work

- Highly sensitive to policy parameters
  - Future work could try online learning of parameters
- Cannot guarantee collision avoidance with target agent
- Not personalized
  - If other agents are humans, they could have their own policy parameters
- Setting the desired velocity
  - Can be set up as an optimization problem to minimize discomfort to the target agent
- Detecting impossible cases
- Incorporating learning of agents' goals

THANK YOU, QUESTIONS?

References

1. J. Van Den Berg, S. J. Guy, M. Lin, and D. Manocha, “Reciprocal n-Body Collision Avoidance,” in Robotics Research, B. Siciliano, O. Khatib, F. Groen, C. Pradalier, R. Siegwart, and G. Hirzinger, Eds.Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, vol. 70, pp.3–19, series Title: Springer Tracts in Advanced Robotics. [Online].Available: http://link.springer.com/10.1007/978-3-642-19457-3 1

2. D. Helbing and P. Molnar, “Social Force Model for PedestrianDynamics,” Phys. Rev. E, vol. 51, no. 5, pp. 4282–4286,May 1995, arXiv:cond-mat/9805244. [Online]. Available: http://arxiv.org/abs/cond-mat/9805244

3. M. Li, M. Kwon, and D. Sadigh, “Influencing leading and followingin human–robot teams,” Autonomous Robots, vol. 45, no. 7, pp.959–978, Oct 2021. [Online]. Available: https://doi.org/10.1007/s10514-021-10016-7

4. M. Kwon, M. Li, A. Bucquet, and D. Sadigh, “Influencing leading andfollowing in human-robot teams,” in Proceedings of Robotics: Scienceand Systems, FreiburgimBreisgau, Germany, June 2019.

5. Y. Gao and C.-M. Huang, “Evaluation of Socially-Aware RobotNavigation,” Front. Robot. AI, vol. 8, p. 721317, Jan. 2022.[Online]. Available: https://www.frontiersin.org/articles/10.3389/frobt.2021.721317/full

6. C. Mavrogiannis, A. M. Hutchinson, J. Macdonald, P. Alves-Oliveira, and R. A. Knepper, ‘Effects of distinct robot navigation strategies on human behavior in a crowded environment’, in 2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI), 2019, pp. 421–430.

