Deploying SICNav in the Field: Safe and Interactive Crowd Navigation using MPC and Bilevel Optimization

Sepehr Samavi[1,2], Garvish Bhutani[2], Florian Shkurti[2], Angela P. Schoellig[1,2]

***Abstract*— Safe and efficient navigation in crowded environments remains a critical challenge for robots that provide a variety of service tasks such as food delivery or autonomous wheelchair mobility. Classical robot crowd navigation methods decouple human motion prediction from robot motion planning, which neglects the closed-loop interactions between humans and robots. This lack of a model for human reactions to the robot plan (e.g. moving out of the way) can cause the robot to get stuck. Our proposed Safe and Interactive Crowd Navigation (SICNav) method is a bilevel Model Predictive Control (MPC) framework that combines prediction and planning into one optimization problem, explicitly modeling interactions among agents. In this paper, we present a systems overview of the crowd navigation platform we use to deploy SICNav in previously unseen indoor and outdoor environments. We provide a preliminary analysis of the system's operation over the course of nearly 7 km of autonomous navigation over two hours in both indoor and outdoor environments.**

I. Introduction

Navigating among human crowds is a critical step towards deploying robots that offer a variety of services, such as autonomous food delivery or wheelchair operation. However, this is a challenging task for robots due to the uncertainty of pedestrian intentions and motions. Conventional approaches use decoupled frameworks (e.g [1], [2]) where human motion prediction (e.g. [3]–[5]) is independent of planning, leading to the robot freezing and potential collisions. Recent interactive strategies (e.g. [6]–[8]) leverage closed-loop frameworks, but must rely on post hoc safety filters to guarantee non-collision, potentially resulting in oscillations or chatter.

This paper presents the initial field deployment of SICNav [9], a bilevel nonlinear MPC framework, which integrates ORCA-modeled human predictions [10] directly into the robot's local planner. SICNav ensures collision-free trajectories while explicitly modeling interactions by jointly optimizing robot plans and human trajectory predictions. Unlike the aforementioned interactive methods, SICNav generates joint robot plans and human predictions that are collision-free by construction, ensuring safety while maintaining interactive behavior.

In this paper, we present the systems overview of the crowd navigation platform we use to deploy SICNav on the Clearpath Jackal robot pictured in Fig. 1. The target task is to place a robot in a priori unknown planar environment with humans and static obstacles, then provide a goal location to the robot and have it navigate to the goal while avoiding collisions with humans and static obstacles.

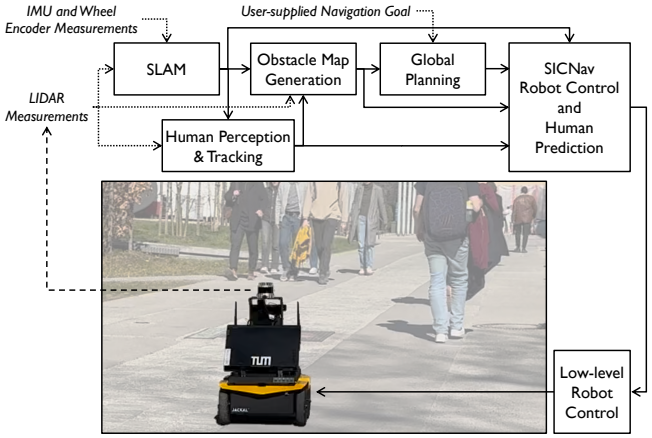


Fig. 1: Photo of the robot operating in a crowd environment superimposed with a block-diagram of the components in the autonomy stack. Video of pictured robot operating autonomously: tiny.cc/sicnav_field_ws

We build upon existing robot autonomy components to enable our robot to navigate in previously unseen indoor and outdoor environments. As illustrated in Fig. 1, our system integrates 2D Simultaneous Localization and Mapping (SLAM) based on Google Cartographer [11], human perception and tracking based on YOLOv9 to generate 2D detections [12] with aUToTrack for human tracking [13], and obstacle map generation and global path planning using the Robot Operating System (ROS) 1 `move_base` framework. Our method, SICNav is used instead of the default local planner in `move_base` to jointly generate robot trajectories and human predictions that are collision-free and interactive with humans. We use the SICNav solutions in receding horizon fashion by sending velocity commands to the robot base, which are tracked by the robot's onboard low-level controllers.

Our contributions are threefold. First, we integrate the aforementioned components of the robot autonomy stack to deploy SICNav in previously unseen indoor and outdoor environments, with noisy state estimation for robot localization, as well as detection and tracking. Second, we propose algorithmic improvements to SICNav and the detection and tracking components to improve the performance of the system. Finally, we provide a preliminary analysis of the system's operation in the field and present our plan to

[1]Learning Systems and Robotics Lab at the Technical University of Munich and Munich Institute for Robotics and Machine Intelligence, Munich, Germany.[2]University of Toronto Robotics Institute and the Vector Institute for Artificial Intelligence, Toronto, Canada. Emails: s.samavi@utoronto.ca, garvish.bhutani@mail.utoronto.ca, florian@cs.toronto.edu, angela.schoellig@tum.de.

Presented at the 2025 IEEE ICRA Workshop on Field Robotics

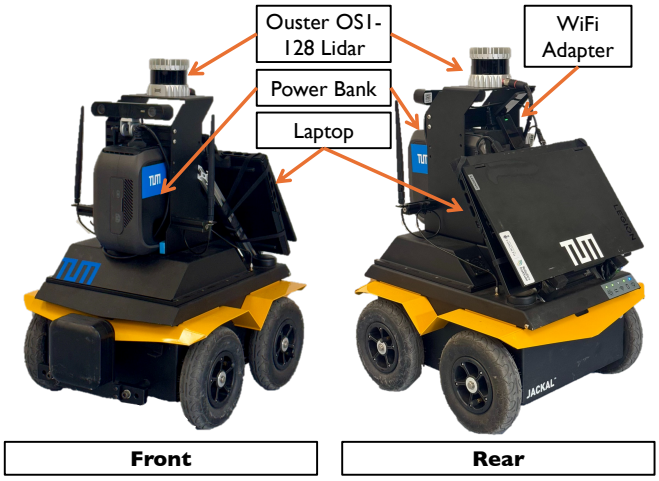


Fig. 2: Photos of the front (left) and rear (right) view of the Clearpath Jackal robot platform with labels for components used in this paper.

thoroughly evaluate the algorithm's performance in the final version of this paper.

II. Methodology

A. The Robot

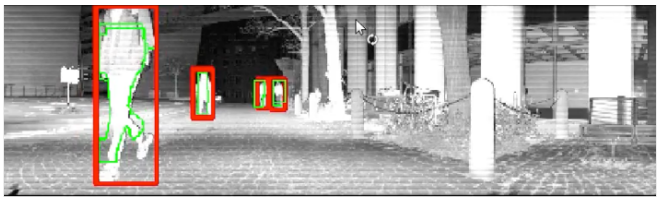
The robot used in our experiments, illustrated in Fig. 2, is a Clearpath Jackal four-wheeled slip steer robot weighing $20kg$ and measuring $46cm$ W $\times 60cm$ L $\times 50cm$ H. We use an Ouster OS1-128 Lidar sensor, its built-in IMU, and the built-in wheel encoders of the robot as the only sensors in this project. The robot is equipped with an onboard computer running Ubuntu 20.04 and ROS 1 Noetic. We additionally mount a laptop with 24-core Intel Core i9-14900HX CPUs and an Nvidia RTX 4080 GPU to the robot.

In order to enable communication with the robot to monitor visualization and issue commands (e.g. specifying goal location, recording data), we use a TP-Link T4U WiFi USB-adapter to broadcast a WiFi network from the laptop. The operator connects to this network using a hand-held laptop or a tablet and uses Virtual Network Computing (VNC) screen sharing to interact with ROS and the RVIZ visualization wirelessly.

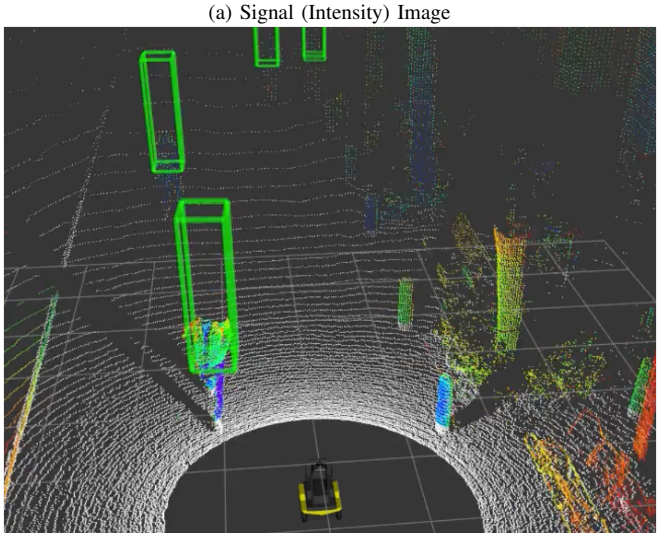
1) Sensor Data Preprocessing: We pre-process the lidar data by employing Patchwork++ [14] to segment ground points from the lidar pointcloud used for mapping and localization as well as obstacle detection. The algorithm proposes a ground likelihood estimation method to segment the ground from the point cloud, tracking the ground plane over time to improve the segmentation. Removing the ground points from the point cloud simplifies our mapping and obstacle detection tasks by reducing the number of points to process.

B. Cartographer SLAM

We employ the 2D version of Google Cartographer [11] for SLAM in our autonomy stack. Our setup relies on lidar point clouds, the lidar's Intertial Measurement Unit (IMU) measurements, and wheel odometry supplied by the robot base. The Cartographer algorithm first collapses the point cloud into a 2D scan in birds-eye-view, using the IMU to estimate the direction of gravity and projecting down. The method then uses scan matching to align incoming lidar scans with generated submaps consisting of a few consecutive scans. To improve scan matching performance, the algorithm incorporates linear and angular velocities from the most recent two odometry readings to help in extrapolating the pose when aligning the incoming lidar scans with the map. To align the submaps into one consistent map, the system constructs a pose graph of all scans and submaps and periodically conducts loop closure optimizations.



(a) Signal (Intensity) Image



(b) Point cCloud

Fig. 3: Lidar signal image (a) and point cloud (b) generated by the Ouster OS1-128 lidar sensor. In (a), we overlay Yolov9 detection bounding boxes (red) and segmentation contours (green). In (b) we illustrate the 3D positions associated with the detections with bounding boxes (green) with the height augmented for easier visualization.

The pose estimation is conducted in the frame of the lidar IMU. We eliminate the need for a lidar-to-IMU calibration by using the lidar's IMU. To transform the odometry data into the IMU frame, we use a static transformation obtained from the Computer Aided Design (CAD) drawings of the robot base. We further filter the pose estimates published by Cartographer using a Extended Kalman Filter (EKF) to estimate the robot's velocity and acceleration which are used in the SICNav optimization problem, described in Sec. II-E.1.

C. Human Detection and Tracking

We adapt the aUToTrack human detection and tracking methodology [13]. The original method obtains 2D bounding boxes on a camera image, then uses a lidar-to-camera calibration to associate the lidar points associated with each bounding box. The method conducts Euclidean clustering to the points associated with each bounding box to obtain a 3D centroid estimate for each bounding box. Since the bounding boxes and lidar points are in different frames, this method is sensitive to poor calibration between the lidar and the camera.

We alter the aUToTrack method by using the signal (measuring intensity), reflectivity, and range (distance) images generated by the Ouster lidar. In these images, each pixel is associated with a point in the point cloud, eliminating the need for a camera altogether. In order to obtain detections, we use the segmentation variant of Yolov9 [12] on the signal and reflectivity images to obtain a contour corresponding to each detection. Fig. 3a illustrates the detections on the intensity image. The contours may contain pixels from the ground or other objects. Thus, in order to reject points that do not belong to the detected object we analyze the range values of the pixels in the detected contour and reject any pixels with range below the 5^{th} percentile and above the 65^{th} percentile of the pixels. From the remaining pixels, we calculate the Euclidean mean of the 3D positions of the detection. Fig. 3b illustrates the positions of the detections in the pointcloud.

Once we obtain a set of detections, we use the aUToTrack method for data association to maintain consistent identities for each detected human through successive frames. We use Kalman Filters (KFs) for with a constant acceleration motion model for each tracked human. Particularly for data association, we use the distance between the predicted state of the track and the 3D position of the detection and follow a greedy data association technique. In order to compensate for tracking delay, we use the constant acceleration model to project the KF estimate forward in time, which we then use in the SICNav optimization problem.

D. Obstacle Map Generation and Global Path Planning

We set up the ROS `move_base` framework with a global and local *Costmap2D*. The framework obtains the map from the Cartographer node then uses lidar measurements to generate a 2D occupancy grid. We filter out point-cloud data associated with moving humans, using their tracks to exclude them from the global map. This process allows SICNav to address the humans within its optimization framework. The global occupancy map is used for global planning with the hybrid A* algorithm, which finds a path for the robot to take to its user-supplied goal by blending discrete search with continuous state evaluations. For the local cost map, we use the ROS 1 *costmap_converter* package [15] to cluster points in the occupancy grid into polygons, which are then translated into line segment representations. We use these segments in SICNav's static obstacle collision avoidance constraints.

E. SICNav

1) Problem Formulation: The environment consists of a robot, human agents, indexed by $j \in \{1, \dots, N\}$, and static obstacles in the form of line segments, indexed by $\tilde{l} \in \{N+1, \dots, N+M\}$. The state of the system is in continuous space and contains the state of the ego robot, and the states of all N humans. The state is denoted as $\mathbf{x}_t = (\mathbf{x}_t^{(r)}, \tilde{\mathbf{x}}_t^{(1)}, \dots, \tilde{\mathbf{x}}_t^{(N)}) \in \mathcal{X}$, where the state of the robot, $\mathbf{x}_t^{(r)}\,(x_t, y_t, \theta_t, v_t, \omega_t, \dot{v}_t, \dot{\omega}_t) \in \mathbb{R}^7$, consists of its 2D position, heading, and longitudinal velocity, angular velocity, longitudinal acceleration, and angular acceleration. The human state, $\tilde{\mathbf{x}}_t^{(j)} \in \mathbb{R}^4$, consists of 2D position, and 2D velocity $\forall j \in \{1, \dots, N\}$.

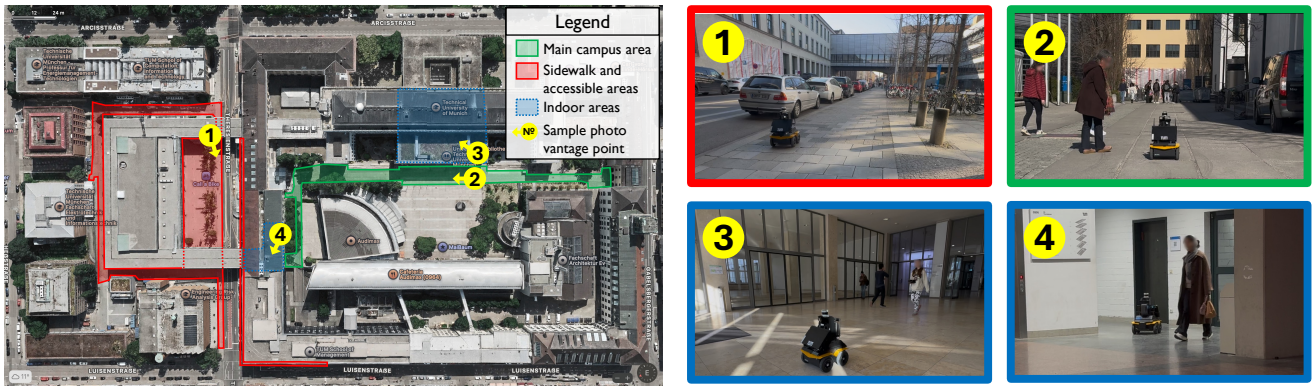
We separate the dynamics of the system into separate functions for the robot and the humans. The robot dynamics, $\mathbf{x}_{t+1}^{(r)} = \mathbf{f}(\mathbf{x}_t^{(r)}, \mathbf{u}_t)$, are modeled as a kinematic unicycle model, where the control input for the system, $\mathbf{u}_t \in \mathbb{R}^2$, is a vector of the linear and angular velocity of the robot for time step t. The dynamics of each human $\forall j \in \{1, \dots, N\}$, $\tilde{\mathbf{x}}_{t+1}^{(j)} = \mathbf{h}(\tilde{\mathbf{x}}_t^{(j)}, \tilde{\mathbf{u}}_t^{(j)})$, are modeled as kinematic integrators, where $\tilde{\mathbf{u}}_t^{(j)}$ denote actions optimal with respect to Optimal Reciprocal Collision Avoidance (ORCA).

2) ORCA for Predicting Human Motion: The ORCA optimization problem for agent j at some time t is a Quadratically Constrained Quadratic Program (QCQP) that solves for a predicted human velocity that minimizes the distance to a intended velocity,

$$\mathcal{O}^{(j)}(\mathbf{x}_t) := \arg\min_{\mathbf{v}} \left\{ \begin{matrix} \|\mathbf{v} - \mathbf{v}_{\text{intent}}(\mathbf{x}_t)\|_2^2 : \\ \text{collision-avoidance} \end{matrix} \right\}, \quad (1)$$

where the optimization variable $\mathbf{v} \in \mathbb{R}^2$ is the velocity of agent j for one time step. The cost is based on a velocity vector from the agent's current position towards the agent's intended goal position, $\mathbf{g}^{(j)}$, which we estimate by projecting the agent's latest observed velocity forward with a fixed time horizon. The non-collision constraints are derived using velocity obstacle approach (see Sec. 4 of [10]) and are composed of a linear constraint for agent j and each other agent $l \neq j$, as well as each static obstacle. These constraints are linear with respect to $\mathbf{v}$ and parameterized by the current state of the system, $\mathbf{x}_t$. The problem also contains a convex quadratic maximum velocity constraint.

The bilevel optimization problem uses the optimum of (1), $\tilde{\mathbf{u}}_t^{(j)}$, to find the refined predicted position of the agent at the next time step, $\tilde{\mathbf{x}}_{t+1}^{(j)}$. This way, if the intended velocity for the agent is feasible with respect to constraints in (1), then the optimal velocity found by solving the problem is the one that moves the agent to the estimated intended position at the next time step.



(a) Map of robot operation areas

(b) Photos of robot operation areas

Fig. 4: (a) Robot operation areas highlighted on the Apple Maps satellite image of the Technical University of Munich campus in Maxvorstadt, Munich, Germany. (b) Photos of the robot operating in each area.

3) SICNav Bilevel Optimization Problem: The bilevel optimization problem is formulated as follows,

$$\underset{\substack{\mathbf{x}_{0:T},\mathbf{u}_{0:T-1},\\ \tilde{\mathbf{u}}_{0:T-1}^{(0:N)}}}{\text{minimize}} \quad \sum_{t=0}^{T-1} l(\mathbf{x}_t,\mathbf{u}_t) + l_T(\mathbf{x}_T) \tag{2a}$$

$$\text{subject to} \quad \mathbf{x}_0 = \mathbf{x}_\text{o} \tag{2b}$$

$$\mathbf{x}_{t+1}^{(r)} = \mathbf{f}(\mathbf{x}_t^{(r)},\mathbf{u}_t) \tag{2c}$$

$$\mathbf{u}_{min}^{(r)} \leq \mathbf{u}_t \leq \mathbf{u}_\text{max}^{(r)} \tag{2d}$$

$$\Delta\mathbf{u}_{min}^{(r)} \leq \mathbf{u}_t - \mathbf{u}_{t-1} \leq \Delta\mathbf{u}_\text{max}^{(r)} \tag{2e}$$

$$\mathbf{x}_t^\top \mathbf{P}_l \mathbf{x}_t \geq {d_l}^2 \tag{2f}$$

$$\tilde{\mathbf{u}}_t^{(j)} \in \mathcal{O}^{(j)}(\mathbf{x}_t) \tag{2g}$$

$$\mathbf{x}_{t+1}^{(j)} = \mathbf{h}(\mathbf{x}_t^{(j)},\tilde{\mathbf{u}}_t^{(j)}) \tag{2h}$$

where the constraints for robot dynamics (2c), robot velocity bounds, (2d) and robot acceleration bounds (2e) are defined for each time step, $\forall t \in \{0,\ldots,T-1\}$; the robot non-collision constraint (2f) is defined for each human agent and each static obstacle in the environment; and the lower-level ORCA optimization problem constraint (2g) and human dynamics constraint (2h) are defined for each human, $\forall j \in \{1,\ldots,N\}$ and each time step. To solve this bilevel problem, we reformulate to a single level by replacing each lower level problem with its Karush-Kuhn-Tucker (KKT) optimality conditions.

We solve the resulting optimization problem using the Acados nonlinear programming solver [16]. We use the solution in receding horizon fashion to generate velocity commands for the robot base, which are tracked by the Jackal's onboard low-level controllers. In the Model Predictive Control (MPC) problem, we use a discrete time step of $\delta\ t\ = 0.25s$, a prediction horizon of $T = 2s$, and we re-plan at $10Hz$. To ensure the problem is solved in real-time, we model the two humans with the lowest time-to-collision with the robot using ORCA. In order to avoid interference from the global planner, we remove the lidar points associated with these humans from the point cloud such that these humans do not appear as obstacles in the costmaps described in Sec. II-D. This way, we treat any additional humans as static obstacles.

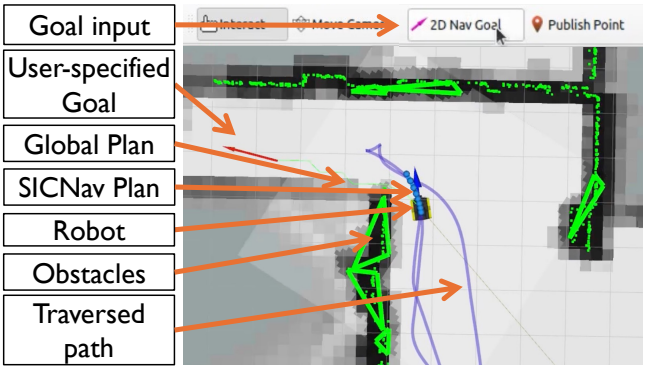


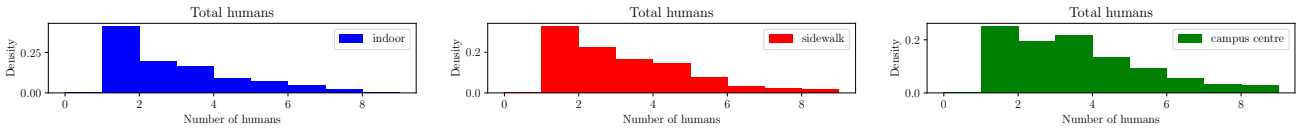
Fig. 5: Top-down view of the Rviz visualization and goal input interface available to the user. The visualization shows a top-down view of the robot's pose in the map that has been constructed using SLAM. In this example the robot is in the operation area labelled 4 in Fig. 4b. The user specifies goal positions a goal to the robot and monitor robot plans.

III. Field Tests

A. Deployment Methodology

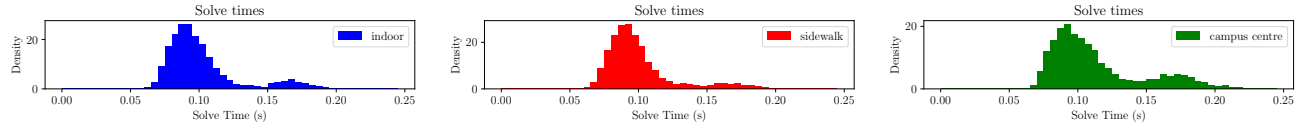
We have begun deploying the robot in the Technical University of Munich campus in Maxvorstadt, Munich, Germany. The campus has a variety of environments, including narrow hallways, open courtyards, and outdoor pathways. The results presented in this paper are preliminary. In Sec. III-C we will highlight the additional field results we plan to include in the final version of the paper.

We have identified three areas for robot operation, as shown in Fig. 4. The *main campus area* (green) is an outdoor pedestrian zone separated from city roads by university buildings. The surfaces in this area are covered variably by cobblestones, asphalt, and pavement. The environment contains a variety of obstacles such as light poles, benches, bicycle parking, and parked cars. The *sidewalk and accessible areas* (red) are the sidewalks and pedestrian zones adjacent to



(a) Indoor areas (b) Sidewalk and Accessible Areas (c) Main Campus Area

Fig. 6: Total number of humans detected in the scene in the different operation areas. We plot histograms for the (a) Indoor areas $\mu = 2.450$, $\sigma = 1.650$; (b) Sidewalk and accessible areas $\mu = 2.716$, $\sigma = 1.796$; and (c) Campus center areas $\mu = 3.128$, $\sigma = 1.986$. We observe that the main campus area had the largest crowds followed by the sidewalk and accessible areas and the indoor areas had the lowest crowds.



(a) Indoor areas (b) Sidewalk and Accessible Areas (c) Main Campus Area

Fig. 7: SICNav solve times in the (a) Indoor areas $\mu = 0.102$, $\sigma = 0.027$; (b) Sidewalk and accessible areas $\mu = 0.100$, $\sigma = 0.026$; and (c) Campus center areas $\mu = 0.114$, $\sigma = 0.035$. All areas had acceptable solve times i.e. lower than discretization period of 0.25. We observe that the indoor and sidewalk areas had similar solve times while the main campus area had larger times.

TABLE I: Autonomous Robot Operation Statistics

	Campus	Sidewalk	Indoor	Total
Duration (h:m:s)	0:57:26	0:37:04	0:17:39	1:51:41
Distance (km)	3.75	2.02	0.96	6.73
# Manual TOs	20	27	2	49
TO Freq. (s^{-1})	0.0058	0.0121	0.0019	0.0073

university buildings and accessible directly adjacent to city roads. This area is similar to the main campus area but additionally contains moving cars on the road adjacent to the sidewalk. The *indoor areas* (blue) are located within two university buildings. The surfaces of the indoor areas are stone flooring with some carpets. The environment contains narrow hallways, staircases, and open spaces with benches and other furniture.

To operate the robot once it is in one of the operation areas, we connect wirelessly to the robot laptop using a hand-held laptop or tablet. Then we view the map that has been collected by the robot (as described in Sec. II-B) in Rviz, as illustrated in Fig. 5. To commence navigation, we specify a goal pose on the map using the 2D Nav Goal interface. Once the robot reaches the pose, it stops operating.

B. Results

Table I summarizes the robot's operation statistics in the three areas. The robot has navigated autonomously for nearly one hour and 51 minutes, traversing 6.73 km across the three environments. A video of the robot navigating in the three areas can be found at `tiny.cc/sicnav_field_ws`.

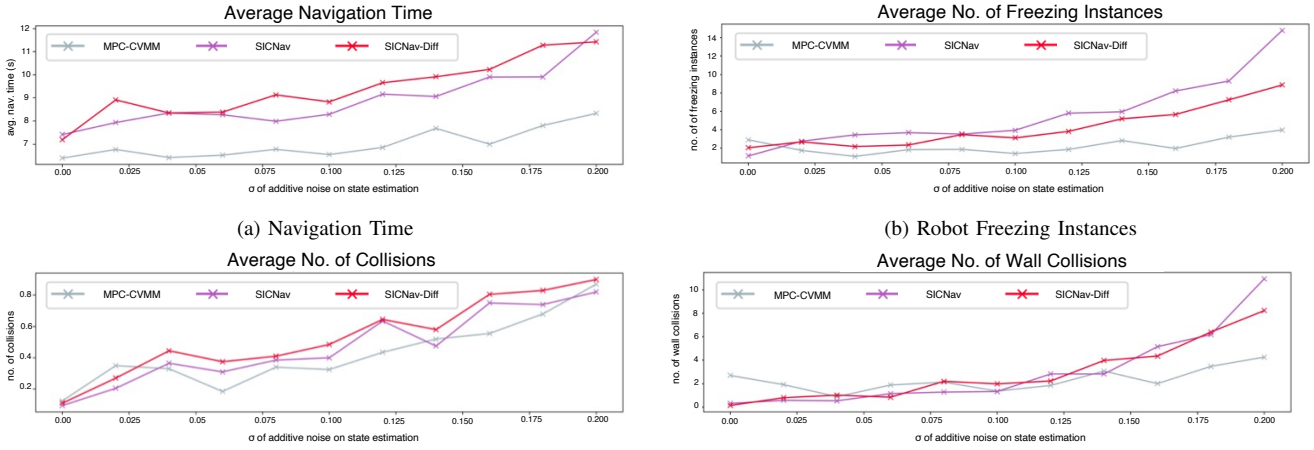
To evaluate robot performance, we first focus on manual takeovers. The sidewalk environment had the highest number of takeovers, totaling 27, with a frequency of 0.0121 s^{-1}. We believe this increase is due to the current setup's inability to detect steps going down, causing the robot to sometimes attempt driving off the sidewalk and requiring manual intervention. In contrast, the campus and indoor environments had fewer takeovers, with frequencies of 0.0058 s^{-1} and 0.0019 s^{-1}, respectively.

We also evaluate the number of humans detected by the robot during operation. Fig. 6 shows the total number of humans detected in the different operation areas. The main campus area had the largest crowds, followed by the sidewalk and accessible areas, while the indoor areas had the lowest crowds. The robot detected an average of 2.45 humans in the indoor areas, 2.72 humans in the sidewalk and accessible areas, and 3.13 humans in the main campus area.

Finally, we evaluate the solve-times of the SICNav optimization problem in the different areas, summarized in Fig. 7. We observe that the indoor and sidewalk areas have similar solve times, while the main campus area had larger times. We believe this is as a result of the higher number of humans encountered in the problem. All areas had acceptable solve times, lower than the discretization period described of 0.25 seconds used in the SICNav MPC problem (see Sec II-E.3).

C. Current Limitations and Next Steps

1) SICNav Performance Comparison with Other Methods: We have so far demonstrated the operation of SICNav in the field. However, in the final version of this paper, we aim to compare the performance of SICNav with other local planning methods in terms of navigation efficiency metrics such as time to goal, distance traveled, and number of manual takeovers. The version of Safe and Interactive Crowd Navigation (SICNav) that we have deployed uses the velocity estimate of the human to predict the intended direction of the human, then uses the ORCA method to predict the humans motion towards this intent while jointly solving for the robot plan. The first method we intend to compare with is SICNav-Diffusion [17] which uses a data-driven diffusion model to predict human intentions and incorporates these intention predictions into the optimization problem. We also intend



(a) Navigation Time

(b) Robot Freezing Instances

(c) Robot-Human Collisions

(d) Robot-Wall Collisions

Fig. 8: Robot crowd navigation performance using different control methods in the face of increasing state estimation noise in simulation. We plot average (a) navigation time, (b) robot freezing instances, (c) number of robot-human collisions, and (d) robot wall collisions over 200 scenarios. When the standard deviation of additive noise increases beyond 0.03 the differences between the methods decreases.

to compare the method with a non-interactive MPC problem that uses a Constant Velocity Motion Model (CVMM) to predict human trajectories without modelling interactions during robot planning.

In previous in-lab real-robot experiments [9], [17] we have shown that SICNav outperforms the non-interactive MPC problem that uses a CVMM to predict human trajectories in terms of metrics measuring robot navigation efficiency and safety. While in the in-lab experiments localization and perception components of the autonomy stack were provided by a VICON motion tracking system, in the field deployment, the robot operates with noisy localization and perception. One question that we aim to answer in the final paper is whether the differences in performance between SICNav and the other methods remain in the face of noisy localization and perception.

We have conducted a preliminary analysis of the relative performance of the methods under noisy localization and perception in simulation. We simulate a 1.75m wide corridor with $N = 3$ humans. We generate 200 scenarios with different initial and final positions on opposite sides of the corridor for each human. We evaluate three methods: SICNav, SICNav-Diffusion [17], and the non-interactive MPC problem with CVMM predictions. To evaluate performance we use average navigation time for the robot to reach its goal over the 200 scenarios, number of instances of robot freezing (where the robot stops moving), number of collisions with humans, and number of collisions with walls. We repeat the 200 scenarios with varying levels of noise added to the robot position, robot velocity, human positions, human velocities, and static obstacle positions. We sample the noise from a zero-mean Gaussian distribution with standard deviation $\sigma \in \{0, 0.2\}$. We then evaluate the performance of the three methods under the noisy conditions.

The results are shown in Fig. 8. In Figs. 8a and 8c we observe that as the standard deviation of the noise increases beyond 0.03, the relative performance of the methods begins changing, indicating that the noise in state estimation does not affect all the evaluated methods in the same way. In the final version of the paper, we aim to conduct a more thorough analysis of the performance of the methods under noisy conditions by varying the noise levels for each component of the autonomy stack separately and also incorporating delays in the perception and localization components of the autonomy stack.

2) Additional Autonomy Components: First, the robot's inability to detect steps going down other non-drivable areas (e.g. grass, traffic lanes) is a significant limitation that we aim to address by incorporating semantic segmentation of the lidar point cloud [18] into the robot autonomy stack. This way we can detect non-drivable areas in the point cloud and include them as obstacles in the costmaps described in section II-D.

Second, In the current setup, if the user intends for the robot to drive to a location that is not currently in the robot's collected map, the user is required to manually specify an intermediate goal, then as the robot makes progress, update the goal once the final destination has appeared in the map. We aim to explore two options to address this limitation. First, we intend to explore collecting a map of the operation area of the robot prior to deployment, then accurately localize inside the map using the existing lidar SLAM method that we currently use for SLAM. If this approach does not scale to the large operating areas of the robot we then aim to explore incorporating a high-level task localization and planning component in addition to the current SLAM and global planning components described in Sec. II-B and Sec. II-D. In the high-level localization module, we intend to use GPS for rough localization on a pre-collected map (e.g. Google Maps) and then use the lidar SLAM method to localize the robot more accurately locally in the map. The high-level planning module will then use the robot's current pose in the map to

generate waypoints to the goal location.

3) Multi-city Testing: So far, we have only tested the robot in the Technical University of Munich campus in Maxvorstadt, Munich, Germany. In the final version of this paper, we aim to test the robot in additional environments such as the city center of Munich, Germany, and the Technical University of Munich campus in Garching, Germany. We also have a similar robot base in the University of Toronto, Toronto, Canada. The second robot uses a lower-end lidar sensor (Ouster OS1-64) and lower-end on-board computer. We aim to test the robot in the University of Toronto campus and the city center of Toronto, Canada to evaluate the generalizability of the robot's performance in different environments and with different hardware configurations.

IV. Conclusion

In this paper we presented the deployment of SICNav, our interactive crowd navigation framework based on bilevel optimization, on a Clearpath Jackal robot in and around the Technical University of Munich campus in Maxvorstadt, Munich, Germany. The robot has autonomously navigated for nearly two hours across three different environments, traversing 6.73 km over nearly two hours of autonomous operation. We demonstrated the ability of our robot to navigate in the face of noisy localization and perception in the field, and we presented a preliminary analysis of the performance of the robot under noisy conditions in simulation. In the final version of this paper, we aim to conduct a more thorough analysis of the performance of the robot under noisy conditions and compare the performance of SICNav with other local planning methods in terms of navigation efficiency metrics such as time to goal, distance traveled, and number of manual takeovers.

References

[1] D. Mayne, J. Rawlings, C. Rao, and P. Scokaert, "Constrained model predictive control: Stability and optimality," *Automatica*, vol. 36, no. 6, pp. 789–814, June 2000. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S0005109899002149

[2] N. E. Du Toit and J. W. Burdick, "Robot Motion Planning in Dynamic, Uncertain Environments," *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 101–115, Feb. 2012. [Online]. Available: doi.org/10.1109/TRO.2011.2166435

[3] T. Salzmann, B. Ivanovic, P. Chakravarty, and M. Pavone, "Trajectron++: Dynamically-Feasible Trajectory Forecasting With Heterogeneous Data," in *2020 European Conference on Computer Vision (ECCV)*, 2020. [Online]. Available: http://arxiv.org/abs/2001.03093

[4] K. Mangalam, Y. An, H. Girase, and J. Malik, "From Goals, Waypoints & Paths To Long Term Human Trajectory Forecasting," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Montreal, QC, Canada: IEEE, Oct. 2021, pp. 15 213–15 222. [Online]. Available: https://ieeexplore.ieee.org/document/9709992/

[5] J. Yue, D. Manocha, and H. Wang, "Human Trajectory Prediction via Neural Social Physics," in *Proceedings of the European Conference on Computer Vision (ECCV)*. arXiv, July 2022, arXiv:2207.10435 [cs]. [Online]. Available: http://arxiv.org/abs/2207.10435

[6] M. Sun, F. Baldini, P. Trautman, and T. Murphey, "Move Beyond Trajectories: Distribution Space Coupling for Crowd Navigation," in *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation, July 2021. [Online]. Available: doi.org/10.15607/RSS.2021.XVII.053

[7] C. Chen, S. Hu, P. Nikdel, G. Mori, and M. Savva, "Relational Graph Learning for Crowd Navigation," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA: IEEE, Oct. 2020, pp. 10 007–10 013. [Online]. Available: https://ieeexplore.ieee.org/document/9340705/

[8] M. Everett, Y. F. Chen, and J. P. How, "Collision avoidance in pedestrian-rich environments with deep reinforcement learning," *IEEE Access*, vol. 9, pp. 10 357–10 377, 2021.

[9] S. Samavi, J. R. Han, F. Shkurti, and A. P. Schoellig, "SICNav: Safe and Interactive Crowd Navigation Using Model Predictive Control and Bilevel Optimization," *IEEE Transactions on Robotics*, vol. 41, pp. 801–818, 2024. [Online]. Available: http://sepehr.fyi/projects/sicnav

[10] 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

[11] W. Hess, D. Kohler, H. Rapp, and D. Andor, "Real-time loop closure in 2d lidar slam," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1271–1278.

[12] C.-Y. Wang and H.-Y. M. Liao, "Yolov9: Learning what you want to learn using programmable gradient information," 2024.

[13] K. Burnett, S. Samavi, S. L. Waslander, T. D. Barfoot, and A. P. Schoellig, "aUToTrack : A Lightweight Object Detection and Tracking System for the SAE AutoDrive Challenge," in *Conference on Computer and Robot Vision (CRV)*, 2019. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/8781613

[14] S. Lee, H. Lim, and H. Myung, "Patchwork++: Fast and robust ground segmentation solving partial under-segmentation using 3d point cloud," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 13 276–13 283.

[15] C. Rosmann, "costmap_converter ros package," 2015. [Online]. Available: https://wiki.ros.org/costmap_converter

[16] R. Verschueren, G. Frison, D. Kouzoupis, J. Frey, N. van Duijkeren, A. Zanelli, B. Novoselnik, T. Albin, R. Quirynen, and M. Diehl, "acados – a modular open-source framework for fast embedded optimal control," *Mathematical Programming Computation*, Oct 2021. [Online]. Available: https://doi.org/10.1007/s12532-021-00208-8

[17] S. Samavi, A. Lem, F. Sato, S. Chen, Q. Gu, K. Yano, A. P. Schoellig, and F. Shkurti, "Sicnav-diffusion: Safe and interactive crowd navigation with diffusion trajectory predictions," *arXiv preprint arXiv:2503.08858*, 2025.

[18] X. Lai, Y. Chen, F. Lu, J. Liu, and J. Jia, "Spherical transformer for lidar-based 3d recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 17 545–17 555.