Enhancing Efficiency and Safety Through Autonomous Navigation in the Intensive Care Unit

Valerio Ponzi[1,2], Adriano Puglisi[1]

[1]Department of Computer, Control and Management Engineering, Sapienza University of Rome, Via Ariosto 25, Roma, 00185, Italy
[2]Institute for Systems Analysis and Computer Science, Italian National Research Council, Via dei Taurini 19, Roma, 00185, Italy

Abstract

The use of autonomous systems in Intensive Care Units (ICUs) has become incredibly important, especially during the COVID-19 pandemic. This period has overwhelmed both ICUs and hospitals, halting many other medical activities and causing significant challenges. This project aims to develop a navigation system tailored specifically for the ICU environment, adapting it to the unique procedures and regulations of that setting. Due to the critical conditions of ICU patients, strict rules dictate precise requirements for navigation, necessitating a context-specific approach. This work will propose a comprehensive navigation system capable of safely guiding from point A to point B within an ICU while addressing the critical issues present in such environments. Unlike traditional Nav2 systems, it will feature specialized collision avoidance components designed specifically for ICU settings, taking into account both contextual demands and the chosen approach. This will involve implementing a multilayered protection technique and employing active movements to prevent collisions with dynamic obstacles.

Keywords

Autonomous navigation, Collision avoidance, Artificial Intelligence, Reinforcement Learning

1. Introduction

Robotic systems play a vital role in dangerous environments and risky situations where human presence is limited. The COVID-19 pandemic has emphasized the necessity for hospitals to adopt automated robotic solutions, particularly in treating patients with highly contagious diseases. These systems minimize the risk of infection for medical staff, addressing the challenges encountered during the early stages of the pandemic. This project aims to develop a navigation system specifically designed for intensive care units (ICUs). It will generate dynamic paths, adhere to contextual rules, avoid collisions with swiftly moving individuals, and effectively manage potentially critical situations.

The challenge of navigation within confined environments is commonly addressed and resolved using established solutions like ROS (Robot Operating System) [49, 32] and Nav2 [48, 35]. ROS, introduced in 2007 through collaborative efforts among universities and robotics-focused companies, has become a standard framework for navigation and robotic applications. It offers modular components and user-friendly monitoring and simulation tools, making it the standard solution for navigation tasks. The extensive contributions and vast library of modular components associated with ROS and Nav2 have solidified their status as essential tools for addressing navigation challenges. Not all tasks can be addressed solely using ROS components, particularly in contexts where specific constraints are required, necessitating the creation of custom nodes. One of the most critical issues in ICU navigation is collision avoidance for dynamic objects. While objects within the intensive care unit are typically stationary, situations may arise where healthcare personnel urgently need to reach a patient's bedside during emergencies. Therefore, it's essential for the robot to navigate while avoiding collisions and without impeding the path of doctors who need to reach the patient promptly.

In addressing navigation challenges within intensive care unit (ICU) environments, researchers have proposed several approaches outlined in the literature. One such method involves the utilization of potential cost maps to navigate around crowded areas [47, 33, 54, 34]). This strategy allows robots to map out regions of high congestion and adjust their paths accordingly to avoid potential collisions. Another critical aspect is collision avoidance for dynamic objects within the ICU. Given the unpredictable nature of medical emergencies, it is essential for robotic systems to adaptively maneuver around moving obstacles to ensure the safety of both patients and medical personnel. Lastly, researchers have explored the concept of employing critical profiles to modulate robotic behavior based on the level of urgency or criticality encountered in the ICU environment, as proposed in the work by [50]. This approach enables robots to dynamically adjust their navigation strategies in response to changing conditions, such as the need for medical professionals to swiftly access patients during emergencies. By

ICYRIME 2023: 8th International Conference of Yearly Reports on Informatics, Mathematics, and Engineering. Naples, July 28-31, 2023
ponzi@diag.uniroma1.it (V. Ponzi); puglisi@diag.uniroma1.it (A. Puglisi)

integrating these diverse approaches, robotic navigation systems can enhance safety and efficiency within ICU settings, ultimately contributing to improved patient care outcomes [9, 38, 8].

These supplementary nodes enable robots to navigate safely through swiftly moving humans. The navigation system can leverage the same framework as proposed by Nav2 but with added enhancements. In navigation systems, route planning is facilitated through a cost map that assigns a cost to each point in the environment, aiding in obstacle identification and the generation of the most efficient trajectory. By integrating a cost map derived from high-traffic areas, it becomes feasible to chart routes that circumvent locations frequented by medical personnel. This approach enhances navigation efficiency and minimizes the likelihood of interference with healthcare professionals.

2. ICU Specific Problems

In designing a navigation system, it's crucial to outline all requirements, constraints, and assumptions pertinent to the context, particularly within an ICU environment. An Intensive Care Unit (ICU) represents an environment of utmost criticality, as all hospitalized individuals are facing severe clinical conditions posing a real and significant threat to their lives. Consequently, ICUs are meticulously structured to maximize the operational efficiency of medical personnel. Beds and associated equipment are strategically positioned within the room, typically in proximity to essential utilities such as oxygen or UPS outlets, as well as room alarms. Additional items like chairs or desks cannot be accommodated, and access for patients' relatives is restricted, with staff expected to complete their tasks promptly and vacate the room immediately. This protocol gained heightened importance during the COVID-19 pandemic, as prolonged presence in the room increased the risk of infection.

In designing the navigation system, it's imperative to ensure that robot movements prioritize human safety while adhering to the following constraints: Medical personnel trajectories must remain unaltered; the robot must never impede their movement. Obstacles' movements are unpredictable, requiring reactive adjustments to detected trajectories. In critical situations where no action can resolve potential collision, the robot must halt immediately, allowing humans to evade collision. Upon the cessation of a dangerous event or potential collision, the robot must revert to its initial goal, either by returning to the initial path, reaching the destination, or devising a new path. Navigation interruption or updating the final goal point should be feasible. Lastly implementing a maximum speed limit for the robot in all situations ensures safety.

3. Related Works

Based on the requirements specified in the previous section, it is possible to identify 2 macro-areas of study which, although strongly related to each other, are 2 topics addressed by previous works independently; they are collision avoidance of dynamic objects and dynamic object identification.

3.1. Collision Avoidance in Navigation

The issue of avoiding collisions while navigating in confined spaces is extensively studied from various angles. Dealing with stationary objects has been thoroughly explored, and numerous algorithms ready for use are available in robotic systems like ROS. However, navigating around moving objects remains a challenge due to its close ties with specific contexts and constraints. Unlike stationary objects, dynamic ones demand a different approach. Their unpredictable nature necessitates swift responses to prevent collisions, and predicting their future positions in the near term could prove invaluable for anticipating and avoiding potential collisions. Multiagent collision avoidance during navigation can be seen as a cooperative task where each agent plays a role in avoiding collisions [42, 36]. In environments like ICU (Indoor Closed Environments), communication between agents is often absent, with sensor data being the only source of information about positions and velocities [10, 1, 40]. Understanding this task involves considering two main approaches: trajectory-based and reaction-based methods [41]. The reaction-based approach focuses on short-term responses, while the trajectory-based method plans over a longer duration. While the latter yields smoother trajectories, it can be computationally intensive. Despite this classification, there's potential in combining these approaches. By leveraging Reinforcement Learning (RL), computationally expensive operations can be moved to offline training stages, with an online policy efficiently queried during runtime. This hybrid approach offers advantages from both methods. Everett et al. propose an RL-based solution aimed at determining velocity vectors to reach goal positions swiftly while avoiding collisions with other agents [42, 17]. Their method employs an LSTM network capable of handling various numbers of agents. Interestingly, the LSTM is used not in a sequential time-based manner, but to encode each agent sequentially, utilizing the final hidden layer for subsequent steps. This design ensures consistent dimensions regardless of the number of agents involved. Another crucial consideration is the impracticality of applying the trajectory-based method in scenarios involving humans, as predicting human paths is challenging due to unpredictable needs or rules. It's often unclear beforehand where a human might need to go or if they may need to return to their initial position for various reasons. Therefore, collision avoidance systems become valuable only when agents are near the target robot and can be analyzed over a short period, during which their movements can be predicted to some extent. Otherwise, the data becomes too noisy for effective trajectory planning. While RL has been utilized in various studies, it's essential to specify constraints on the behavior of other agents to yield meaningful results, as highlighted by Everett. Generic approaches where each agent exhibits arbitrary behavior may not lead to effective collision avoidance strategies. An alternative approach is presented in [53], which leverages Lidar data to identify dynamic objects, track their movements, and employ the ORCA algorithm [52] to compute collision-free paths. This method offers another avenue for effective collision avoidance in dynamic environments.

Optimal Reciprocal Collision Avoidance (ORCA) [52] stands out as a velocity-based planning technique renowned for its ability to ensure collision avoidance in both static and dynamic environments, boasting high scalability. ORCA evaluates the velocities of all agents involved and delineates cones representing potential collision scenarios. Subsequently, through an optimization process, it determines the minimum velocity necessary to navigate out of these collision cones.

While ORCA offers robust collision avoidance capabilities, it operates on an optimization search paradigm, which implies that it can find a solution multiple times if one exists. However, it does have a couple of drawbacks. Firstly, it assumes homogeneity among objects within the workspace, whether static or following the same navigation policy. Deviating from this assumption can lead to catastrophic outcomes, as noted by Vince Kurtz [44]. Secondly, its reliance on optimization introduces a notable delay, potentially resulting in missed solutions. Such delays could pose challenges in environments requiring swift decision-making.

In response to these limitations, was proposed an alternative approach. This method involves predicting the motion of dynamic objects over a short timeframe using an LSTM RNN with online training, as suggested in [41]. Once these predictions are available, they are integrated into a Nonlinear Probabilistic Velocity Obstacles algorithm. This adapted algorithm effectively handles collision avoidance in static environments and accounts for objects moving along predictable trajectories, derived from short-term predictions.

3.2. Dynamic Object Identification

In navigating around obstacles, whether they are static or dynamic, the first step is to detect them using sensors installed on the robot. Ready-to-use components, such as those found in ROS and Nav2, are commonly used to identify static obstacles within a local map, which is then integrated into the navigation system. Dynamic obstacles require a different approach, as their velocities necessitate fast and specific algorithms capable of calculating both position and velocity. A method proposed by In [43] was introduced as a technique for identifying dynamic objects using spatiotemporal norms derived from points gathered by robotic sensors, such as Lidar or depth cameras. This method involves clustering or creating point clouds from the sensor-detected points, representing potential objects for analysis. Subsequently, a spatiotemporal norm analysis is applied to these point clouds to determine if the objects are undergoing translation or rotation. This approach enables the recovery of an object's position using sensor data. By analyzing the covariance (a measure of sparsity) of the point cloud, possible rotations and the object's radius can be reconstructed, providing valuable information for collision avoidance systems. It's crucial to note that these results are based solely on points identified by the Lidar or depth camera. This means that the central position and covariance calculations are limited to the visible parts of the objects; any points hidden from the sensors are not considered in the analysis. Therefore, this analysis is inherently related to the visible portions of the objects.

4. ICU Navigation System

Navigating an ICU has specific demands that must be satisfied in a robust architecture that is modified and extended for that. The presented navigation system is based on the Nav2 infrastructure in the ROS2 environment with custom components (or nodes) specific to the ICU context. This project has a Gazebo simulation with rViz monitoring. The architecture comprises a standard navigation system supplemented by three additional nodes designed to enhance collision avoidance capabilities.

The proposed system navigates the robot from point A to point B using a cost map generated by SLAM [2]. In an Intensive Care Unit (ICU), where minimizing instrument presence and their impact on medical staff movement is crucial, the map's variability is assumed to be rare. It's created extemporaneously and maintained throughout the robot's operational lifecycle, with manual updates only in exceptional cases. This approach doesn't significantly constrain the project, as static objects not on the map can still be identified and managed by local and collision avoidance components with regularly updated perspectives.

Given Nav2's adeptness at handling navigation in closed environments, this project focuses on collision avoidance, the primary ICU navigation requirement. Avoiding static or dynamic objects and medical personnel is achieved through a multilayered approach, each

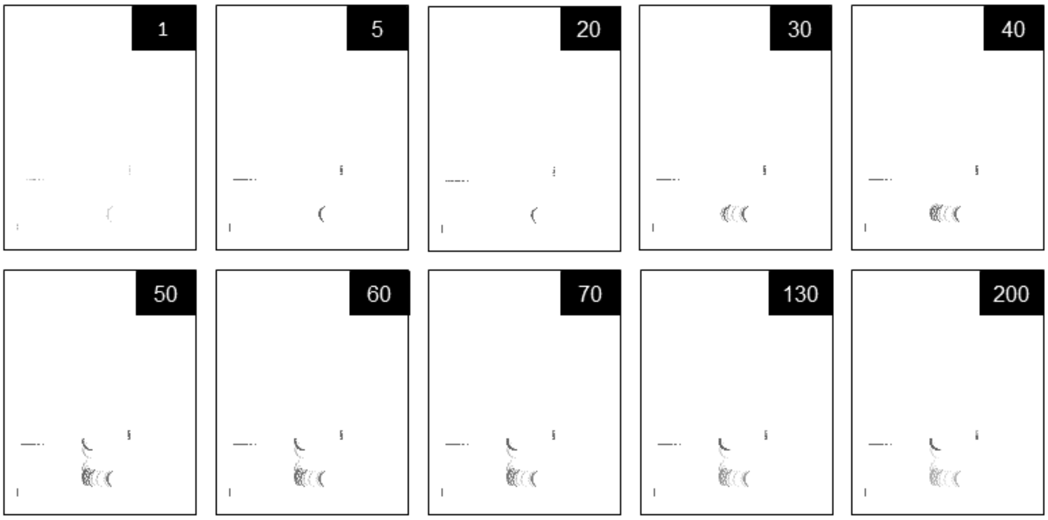

Figure 1: Images showing the evolution of potential values as an object moves, with adjustments made to the decay rate parameter τ influencing the rate of potential decrease over time.

level enhancing security in this task:

- Localmap: Identifies new objects via local maps and adjusts navigation accordingly, effective for static but less so for dynamic obstacles.
- Potential Areas: Modifies the cost map based on the likelihood of encountering obstacles in certain areas, thereby reducing the probability of choosing paths through them.
- Collision Monitor: A ROS2 node that reduces the robot's speed when obstacles are nearby.
- Emergency Guard: Adjusts movement profiles or capabilities based on the distance from obstacles, potentially slowing down or stopping the robot.
- Dynamic Collision Avoidance: Alters the robot's trajectory to avoid collisions dynamically.

Each layer addresses specific scenarios, bolstering safety and meeting ICU requirements effectively.

4.1. Localmap

During navigation, sensor data create a cost map close to the robot with a high-frequency update. This updated and high-quality map is used to adapt the trajectory, avoiding obstacles and keeping the original track (defined by the Planner Server) as much as possible. This node is already present in the Nav2 framework and is used as is, for this reason will not be analyzed anymore.

4.2. Potential Areas

As per the requirements, the navigation system must prioritize avoiding collisions with medical personnel. Hence, it's vital to identify areas with a significant probability of encountering people or obstacles and steer clear of trajectories leading into those areas. In the cost map, potential values range from integers 0 to 255. High values, like 255 and 254 (considered as lethal costs), indicate a high probability of collision with an object, while very low values such as 0 or 1 (representing free space costs) denote no obstacles and safe navigation. Each value signifies a distinct collision probability based on proximity to obstacles. The component identifies contact points of lidar rays with objects (referred to as hit points HP). For every point in the hit point set, it's inferred that an object exists at that point, thus necessitating an increase in its potential. Conversely, if there's no hit (a point not in HP), the potential is decreased (indicating the absence of an object at that point) following an exponential function with decay parameter τ. Figure 1 illustrates how potential values evolve (depicted in the black box in the upper right corner) as an object moves, and how potentials decrease over time. Adjusting τ enables control over the decay rate.

$$\begin{aligned} Ptnl(p, t + \Delta t) = \min(&\text{MAX\_PTNL}; \\ &Ptnl(p, t) * e^{-\tau \Delta t} + \delta(p \in HP)) \end{aligned} \tag{1}$$

In equation 1, the potential is refreshed every Δt. If point p is hit by a ray (in HP), its value increases; otherwise, it decreases gradually over time with a decay period. To maintain flexibility in navigation, the potential is capped (MAX_PTNL), allowing trajectories through those areas albeit with a low probability. Subsequently, the potential map is integrated into the environment map and regularly updated with new potential values (refer to Figure 2).

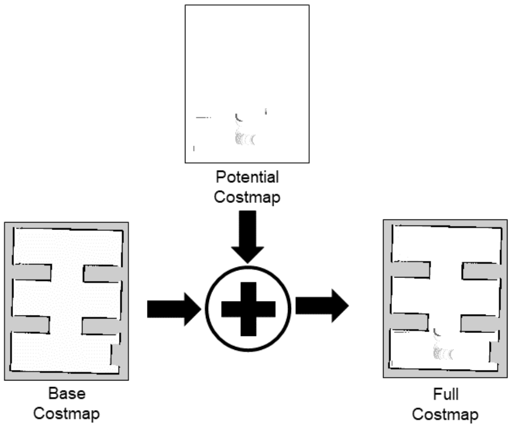

Figure 2: Integration of Potential Map into Environment Map: The process of merging potential values into the environment map, ensuring updated information for safer trajectory planning.

4.3. Collision Monitor

The Collision Monitor, a node available within the Nav2 framework, serves as a crucial safety feature in the navigation system. Positioned just before the command is dispatched to the robot, its primary function is to ensure safe navigation. When the robot approaches an obstacle, the Collision Monitor intervenes by reducing its velocity. Specifically, it may decrease the velocity to a fraction of the original command, such as 20%, to prevent collisions. However, if the robot is not close to any obstacles, the command remains unchanged. Since the Collision Monitor is an integral part of the Nav2 framework and is utilized without modification, it will not be further analyzed as its functionality is standardized and deemed sufficient for the system's safety requirements.

4.4. Emergency Guard

Emergency Guard is a custom ROS2 node that improves the capability of a standard collision monitor. The component identifies the minimum distance with a generic object (both static and dynamic), evaluates the variation compared to the previous detection, and based on that predicts the new minimum distance that will potentially be detected.

$$\hat{d}_{min}(t) = d_{min}(t) + (d_{min}(t) - d_{min}(t-1)) \quad (2)$$

If the predicted distance falls below a warning threshold (THR_WR), the navigation system applies a reduced-speed movement profile. However, if it drops below a critical threshold (THR_CR), the robot transitions into a blocking profile. This blocking profile indicates a highly critical situation, prompting the robot to halt, allowing humans to intervene and avoid a collision.

$$Profile = \begin{cases} Critical & \hat{d}_{min}(t) < THR\_CR; \\ Warning & \hat{d}_{min}(t) < THR\_WR; \\ Normal & otherwise \end{cases} \quad (3)$$

Once the predicted distance exceeds the critical threshold, the robot remains in a critical state until both the warning threshold and normal conditions are met. For safety reasons, the robot maintains its critical state until it returns to a normal operating state. This behavior is depicted in the state diagram shown in Figure 3.

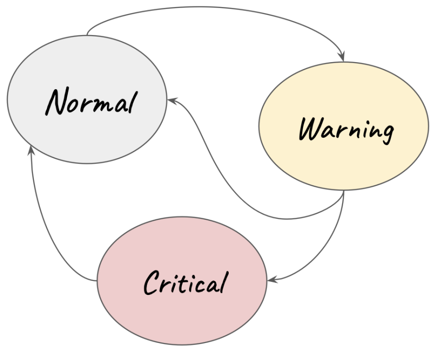

Figure 3: State diagram of emergency guard node

4.5. Dynamic Object Identification

A crucial aspect of collision avoidance is the identification of dynamic objects and the extraction of their positions and velocities. To achieve this, an approach inspired by the work of Raphael Falque [43] has been adapted to suit the specific context. The algorithm encompasses the following steps:

Hit Point Identification: Identifying contact points of laser rays with surrounding objects. Each point represents a hit point on an object.

Environment Filtering: Filtering out all hit points originating from the ICU environment. Since this is focused on dynamic object detection, static objects within the environment are disregarded. Filtering is accomplished using a kNN (k-Nearest Neighbors) algorithm. A kNN model is trained using the environment's cost map. For each hit point, nearby points are identified using the Nearest Neighbors of the trained model. If at least one point is at a critical level (indicating the presence of an environmental object), the hit point is filtered out. This filtering process helps eliminate noise from detection.

Point Cloud Creation: After filtering, the remaining hit points are clustered using a DBScan algorithm. This approach offers the advantage of dividing points into groups without needing to specify the number of clusters beforehand. However, it requires careful tuning to ensure the creation of sparse clusters. This tuning typically involves setting a high value for the epsilon parameter (eps) and a low value for the minimum number of samples parameter (min_sample). The resulting clusters form the point clouds representing the identified objects. To eliminate isolated points resulting from incorrect distance measurements, clusters with a small number of elements are filtered out. Additionally, any points not associated with a cluster are removed from consideration.

Dinamicity Identification: Each point cloud is analyzed to determine its movement characteristics. Initially, the center of mass for the points in point cloud i, denoted as $\mathcal{N}^i$, is computed. This is achieved by calculating the average position of its hit points, as shown in equation 4. Subsequently, the covariance of the point cloud, represented by equation 5, is determined. This covariance provides information about the dispersion of the points around the center of mass, aiding in recognizing the object's movement pattern.

$$m^i = \frac{1}{||\mathcal{N}^i||} \sum_{p^i_j \in \mathcal{N}^i} p^i_j \tag{4}$$

Detecting movement involves assessing significant variations in the center of the point cloud. However, noise in measurements can lead to fluctuations in the center that need to be filtered out. One common approach to identify variations or trends in sequential values is to use linear regression. However, this method is not suitable in this context because rapid generation of points results in a dataset with values that are nearly constant or exhibit very small variations. Consequently, the linear regression yields a zero coefficient with a prediction constant equal to the mean value. Any deviations from this fixed value are interpreted as errors rather than meaningful changes. To address this issue and mitigate the effects of noise and the high number of values, a Naive approach is proposed. This approach reconstructs the area of movement over time, defined as the rectangle encompassing the movement (see Figure 4). If the area exceeds a predefined threshold, dynamic behavior is identified.

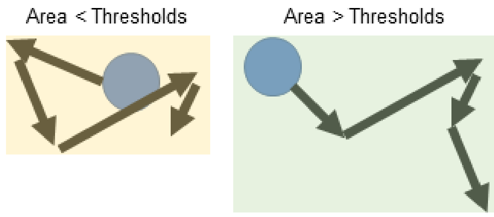

Figure 4: Images depicting the Naive approach's reconstruction of the movement area over time, with dynamic behavior identified based on the threshold-exceeded area

$$cov^i = \frac{1}{||\mathcal{N}^i||} \sum_{p^i_j \in \mathcal{N}^i} (p^i_j - m^i)(p^i_j - m^i)^T \tag{5}$$

When analyzing the variation of covariance, two scenarios may arise: rotation of the object or a reduction in distance between the object and the robot. While theoretically, significant variability in covariance over some time could indicate movement, strong measurement noise can lead to frequent changes in covariance even without real movement. Therefore, covariance analysis is not reliable for evaluating and identifying object dynamics.

Dynamic Object Measurement: Given the list of dynamic objects, their positions (or central positions of points in their cloud) and velocities are recovered from previous points. This information is then utilized by the collision avoidance engine. Calculating velocity by analyzing close positions in time, such as from two consecutive measurements, often results in significant positional errors and speed oscillations. To mitigate this, a period T (or a number X of measurements or odometry messages) is considered to calculate velocity (see equation 6). This approach helps reduce measurement errors and ensures smoother velocity estimation.

$$velocity = \frac{position(t+T) - position(t)}{T} \tag{6}$$

4.6. Dynamic Collision Avoidance

The Collision Avoidance node is designed to prevent collisions with dynamically moving objects that may pose an imminent threat to the robot. Its primary objective is not to define the entire navigation path towards the goal, but rather to handle complex dynamic situations effectively. When a critical situation arises, indicated by

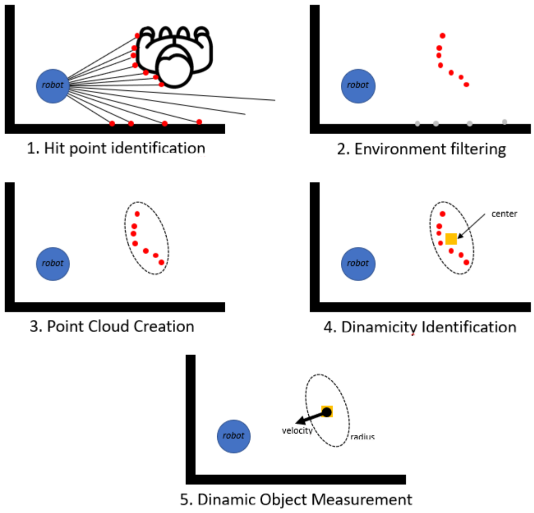

Figure 5: Phases of dynamic object identification

objects being closer than a specific threshold, the node retrieves positional and velocity information of dynamic objects from the dedicated Dynamic Object Identification node (see section 4.5), along with the robot's position and the goal pose. It then applies an algorithm to determine the appropriate speed to resolve the critical event. Once the critical situation is resolved (i.e., objects are no longer too close), the node is deactivated, and standard navigation by the Nav2 Plan Server resumes. This node is based on the Deep Reinforcement Learning strategy proposed by Michael Everett [41]. The algorithm adopts a multi-agent approach capable of handling a variable number of dynamic objects. To accommodate the variable number of obstacles, they are standardized using an LSTM network. Each obstacle's state is fed into an LSTM network, and only the last hidden layer is utilized in subsequent processing (see Figure 6). This condensed representation of dynamic objects, along with the robot's state, is used to generate a vector passed through two fully connected layers, resulting in a probability distribution for possible actions.

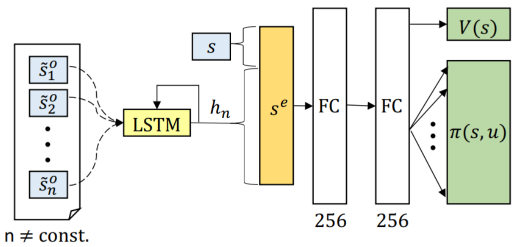

Figure 6: Full network to generate a probability distribution for the actions

5. Implementation

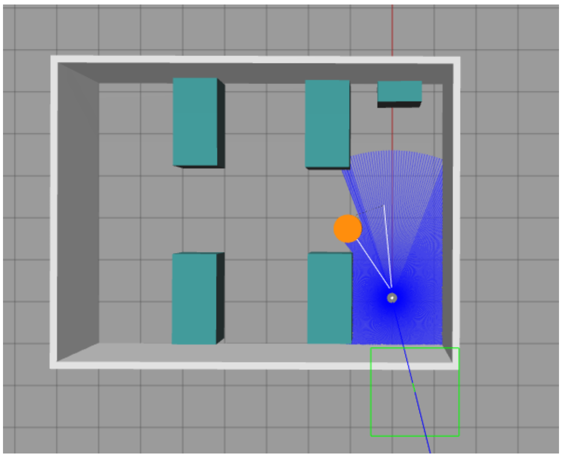

Figure 7: Gazebo simulation of a simple ICU

This project was developed on Ubuntu 22.04 using ROS2 Humble distribution and Nav2. Custom nodes were implemented using Python 3.10 and interfaced with ROS services. Simulations and monitoring were conducted using Gazebo and rViz. The robot and environment were adapted from turtlebot3, with a custom world tailored to resemble an ICU environment (refer to Figure 5).

6. Results

The development process was marked by a series of thorough tests and evaluations, both for individual custom nodes and for the entire system. This approach ensured that each node could complete its task, ensuring safe behavior in the robot's movement. In particular, detailed tests were conducted to explore a range of scenarios, including edge cases, to identify and address any issues. One of the main challenges encountered during development was the presence of noise in measurements, particularly evident in the contact points detected by sensors. These fluctuations in position could compromise the system's reliability, necessitating careful calibration and implementation of advanced filtering algorithms. The goal was to minimize the impact of noise and ensure proper interpretation of data by the system.

Another significant challenge was managing computational resources, critical for the proper functioning of the system, especially in real-time environments like robotics. Insufficient resources could lead to delays in message processing and calculations, potentially affecting overall system performance. Consequently, optimizing resource usage through the implementation of parallelization techniques, optimization algorithms, or potential hardware upgrades was essential.

Ultimately, successfully addressing these challenges was crucial for the project's progress. Through an integrated approach involving thorough testing, algorithm optimization, and resource management, the system was able to ensure reliable behavior in real-world applications.

7. Conclusion

The proposed solution adopts the architectural infrastructure of a widely used navigation system based on Nav2, with customized extensions tailored specifically for the ICU environment. This standardized approach facilitated rapid prototyping and integration within the existing navigation framework.

Custom navigation components designed for the ICU environment contribute to enhanced safety characteristics during navigation. For instance, the Emergency Guard component complements the Collision Monitor by introducing different profiles that gradually restrict the robot's movement freedom. These profiles adjust parameters such as speed or radius, crucial for collision avoidance algorithms, while also incorporating simple prediction mechanisms to preemptively address potential safety risks.

Potential maps play an important role in shaping the costmap based on the likelihood of encountering obstacles, thus enabling the generation of safer trajectories. Careful configuration of potential values, including setting appropriate decay periods, ensures the creation of up-to-date potential maps that accurately reflect recent object detections. Fine-tuning these parameters is essential to strike a balance between maintaining high potential values in areas with recent obstacles while avoiding excessive averaging that could diminish the effectiveness of this feature.

The collision avoidance algorithm prioritizes rapid decision-making through the adoption of an online reinforcement learning (RL) model. This node selectively activates only during critical situations, swiftly deactivating once the threat has passed. Importantly, in scenarios where the collision avoidance movement brings the robot into proximity with static objects, other nodes such as the Collision Monitor or Emergency Guard are triggered to avert collisions and uphold overall safety.

In summary, the integrated system offers a robust solution for navigating within an ICU environment, ensuring a high level of safety through the coordinated efforts of multiple nodes. By addressing a range of potentially critical events with distinct functionalities, the system effectively meets the initial safety requirements of ICU navigation.

References

[1] Y. F. Chen, M. Liu, M. Everett, J. P. How, Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning, in: 2017 IEEE International Conference on Robotics and Automation (ICRA), 2017, pp. 285–292. doi:10.1109/ICRA.2017.7989037.

[2] H. Durrant-Whyte, T. Bailey, Simultaneous localization and mapping: part i, IEEE Robotics & Automation Magazine 13 (2006) 99–110. doi:10.1109/MRA.2006.1638022.

[3] G. Capizzi, S. Coco, G. L. Sciuto, C. Napoli, A new iterative fir filter design approach using a gaussian approximation, IEEE Signal Processing Letters 25 (2018) 1615 – 1619. doi:10.1109/LSP.2018.2866926.

[4] M. Woźniak, D. Połap, C. Napoli, E. Tramontana, Graphic object feature extraction system based on cuckoo search algorithm, Expert Systems with Applications 66 (2016) 20 – 31. doi:10.1016/j.eswa.2016.08.068.

[5] C. Napoli, G. Pappalardo, E. Tramontana, Z. Marszalek, D. Polap, M. Wozniak, Simplified firefly algorithm for 2d image key-points search, 2014. doi:10.1109/CIHLI.2014.7013395.

[6] C. Napoli, G. Pappalardo, E. Tramontana, Improving files availability for bittorrent using a diffusion model, 2014, pp. 191 – 196. doi:10.1109/WETICE.2014.65.

[7] G. Capizzi, F. Bonanno, C. Napoli, Hybrid neural networks architectures for soc and voltage prediction of new generation batteries storage, 2011, pp. 341 – 344. doi:10.1109/ICCEP.2011.6036301.

[8] M. Wozniak, C. Napoli, E. Tramontana, G. Capizzi, G. Lo Sciuto, R. K. Nowicki, J. T. Starczewski, A multiscale image compressor with rbfnn and discrete wavelet decomposition, volume 2015-September, 2015. doi:10.1109/IJCNN.2015.7280461.

[9] M. Woźniak, D. Połap, R. K. Nowicki, C. Napoli, G. Pappalardo, E. Tramontana, Novel approach toward medical signals classifier, volume 2015-September, 2015. doi:10.1109/IJCNN.2015.7280556.

[10] D. Połap, M. Woźniak, C. Napoli, E. Tramontana, R. Damaševičius, Is the colony of ants able to recognize graphic objects?, Communications in Computer and Information Science 538 (2015) 376 – 387. doi:10.1007/978-3-319-24770-0_33.

[11] M. Woźniak, D. Połap, M. Gabryel, R. K. Nowicki, C. Napoli, E. Tramontana, Can we process 2d images using artificial bee colony?, volume 9119, 2015, pp. 660 – 671. doi:10.1007/978-3-319-19324-3_59.

[12] B. A. Nowak, R. K. Nowicki, M. Woźniak, C. Napoli,

Multi-class nearest neighbour classifier for incomplete data handling, volume 9119, 2015, pp. 469 – 480. doi:10.1007/978-3-319-19324-3_42.
[13] C. Napoli, G. Pappalardo, E. Tramontana, R. K. Nowicki, J. T. Starczewski, M. Woźniak, Toward work groups classification based on probabilistic neural network approach, volume 9119, 2015, pp. 79 – 89. doi:10.1007/978-3-319-19324-3_8.
[14] G. Borowik, M. Woźniak, A. Fornaia, R. Giunta, C. Napoli, G. Pappalardo, E. Tramontana, A software architecture assisting workflow executions on cloud resources, International Journal of Electronics and Telecommunications 61 (2015) 17 – 23. doi:10.1515/eletel-2015-0002.
[15] F. Bonanno, G. Capizzi, G. L. Sciuto, C. Napoli, Wavelet recurrent neural network with semiparametric input data preprocessing for micro-wind power forecasting in integrated generation systems, 2015, pp. 602 – 609. doi:10.1109/ICCEP.2015.7177554.
[16] G. Capizzi, G. L. Sciuto, C. Napoli, E. Tramontana, A multithread nested neural network architecture to model surface plasmon polaritons propagation, Micromachines 7 (2016). doi:10.3390/mi7070110.
[17] D. Połap, M. Woźniak, C. Napoli, E. Tramontana, Real-time cloud-based game management system via cuckoo search algorithm, International Journal of Electronics and Telecommunications 61 (2015) 333 – 338. doi:10.1515/eletel-2015-0043.
[18] F. Bonanno, G. Capizzi, S. Coco, C. Napoli, A. Laudani, G. L. Sciuto, Optimal thicknesses determination in a multilayer structure to improve the spp efficiency for photovoltaic devices by an hybrid fem - cascade neural network based approach, 2014, pp. 355 – 362. doi:10.1109/SPEEDAM.2014.6872103.
[19] G. Capizzi, F. Bonanno, C. Napoli, A wavelet based prediction of wind and solar energy for long-term simulation of integrated generation systems, 2010, pp. 586 – 592. doi:10.1109/SPEEDAM.2010.5542259.
[20] F. Bonanno, G. Capizzi, C. Napoli, Some remarks on the application of rnn and prnn for the charge-discharge simulation of advanced lithium-ions battery energy storage, 2012, pp. 941 – 945. doi:10.1109/SPEEDAM.2012.6264500.
[21] F. Bonanno, G. Capizzi, G. L. Sciuto, C. Napoli, G. Pappalardo, E. Tramontana, A novel cloud-distributed toolbox for optimal energy dispatch management from renewables in igss by using wrnn predictors and gpu parallel solutions, 2014, pp. 1077 – 1084. doi:10.1109/SPEEDAM.2014.6872127.
[22] C. Napoli, G. Pappalardo, E. Tramontana, A hybrid neuro-wavelet predictor for qos control and stability 8249 LNAI (2013) 527 – 538. doi:10.1007/978-3-319-03524-6_45.
[23] G. Capizzi, G. L. Sciuto, C. Napoli, M. Woźniak, G. Susi, A spiking neural network-based long-term prediction system for biogas production, Neural Networks 129 (2020) 271 – 279. doi:10.1016/j.neunet.2020.06.001.
[24] C. Napoli, G. Pappalardo, E. Tramontana, Using modularity metrics to assist move method refactoring of large systems, 2013, pp. 529 – 534. doi:10.1109/CISIS.2013.96.
[25] F. Bonanno, G. Capizzi, G. L. Sciuto, C. Napoli, G. Pappalardo, E. Tramontana, A cascade neural network architecture investigating surface plasmon polaritons propagation for thin metals in openmp 8467 LNAI (2014) 22 – 33. doi:10.1007/978-3-319-07173-2_3.
[26] F. Bonanno, G. Capizzi, A. Gagliano, C. Napoli, Optimal management of various renewable energy sources by a new forecasting method, 2012, pp. 934 – 940. doi:10.1109/SPEEDAM.2012.6264603.
[27] G. Capizzi, G. L. Sciuto, C. Napoli, R. Shikler, M. Wozniak, Optimizing the organic solar cell manufacturing process by means of afm measurements and neural networks, Energies 11 (2018). doi:10.3390/en11051221.
[28] G. Capizzi, C. Napoli, L. Paternò, An innovative hybrid neuro-wavelet method for reconstruction of missing data in astronomical photometric surveys 7267 LNAI (2012) 21 – 29. doi:10.1007/978-3-642-29347-4_3.
[29] S. Russo, C. Napoli, A comprehensive solution for psychological treatment and therapeutic path planning based on knowledge base and expertise sharing, volume 2472, 2019, pp. 41 – 47.
[30] N. N. Dat, V. Ponzi, S. Russo, F. Vincelli, Supporting impaired people with a following robotic assistant by means of end-to-end visual target navigation and reinforcement learning approaches, volume 3118, 2021, pp. 51 – 63.
[31] V. Marcotrigiano, G. D. Stingi, S. Fregnan, P. Magarelli, P. Pasquale, S. Russo, G. B. Orsi, M. T. Montagna, C. Napoli, C. Napoli, An integrated control plan in primary schools: Results of a field investigation on nutritional and hygienic features in the apulia region (southern italy), Nutrients 13 (2021). doi:10.3390/nu13093006.
[32] V. Ponzi, S. Russo, V. Bianco, C. Napoli, A. Wajda, Psychoeducative social robots for an healthier lifestyle using artificial intelligence: a case-study, volume 3118, 2021, pp. 26 – 33.
[33] G. De Magistris, R. Caprari, G. Castro, S. Russo, L. Iocchi, D. Nardi, C. Napoli, Vision-based holistic scene understanding for context-aware human-robot interaction 13196 LNAI (2022) 310 – 325. doi:10.1007/978-3-031-08421-8_21.

[34] N. Brandizzi, S. Russo, G. Galati, C. Napoli, Addressing vehicle sharing through behavioral analysis: A solution to user clustering using recency-frequency-monetary and vehicle relocation based on neighborhood splits, Information (Switzerland) 13 (2022). doi:10.3390/info13110511.

[35] S. Pepe, S. Tedeschi, N. Brandizzi, S. Russo, L. Iocchi, C. Napoli, Human attention assessment using a machine learning approach with gan-based data augmentation technique trained using a custom dataset, OBM Neurobiology 6 (2022). doi:10.21926/obm.neurobiol.2204139.

[36] A. Alfarano, G. De Magistris, L. Mongelli, S. Russo, J. Starczewski, C. Napoli, A novel convmixer transformer based architecture for violent behavior detection 14126 LNAI (2023) 3 – 16. doi:10.1007/978-3-031-42508-0_1.

[37] C. Ciancarelli, G. De Magistris, S. Cognetta, D. Appetito, C. Napoli, D. Nardi, A gan approach for anomaly detection in spacecraft telemetries 531 LNNS (2023) 393 – 402. doi:10.1007/978-3-031-18050-7_38.

[38] I. E. Tibermacine, A. Tibermacine, W. Guettala, C. Napoli, S. Russo, Enhancing sentiment analysis on seed-iv dataset with vision transformers: A comparative study, 2023, pp. 238 – 246. doi:10.1145/3638985.3639024.

[39] G. D. Magistris, C. Rametta, G. Capizzi, C. Napoli, Fpga implementation of a parallel dds for wide-band applications, volume 3092, 2021, pp. 12 – 16.

[40] G. De Magistris, M. Romano, J. Starczewski, C. Napoli, A novel dwt-based encoder for human pose estimation, volume 3360, 2022, pp. 33 – 40.

[41] M. Everett, Y. F. Chen, J. P. How, Motion planning among dynamic, decision-making agents with deep reinforcement learning, in: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 2018. URL: https://arxiv.org/pdf/1805.01956.pdf.

[42] M. Everett, Y. F. Chen, J. P. How, Collision avoidance in pedestrian-rich environments with deep reinforcement learning, IEEE Access 9 (2021) 10357–10377. doi:10.1109/ACCESS.2021.3050338.

[43] R. Falque, C. L. Gentil, F. Sukkar, Dynamic object detection in range data using spatiotemporal normals, 2023. arXiv:2310.13273.

[44] V. Kurtz, H. Lin, Toward verifiable real-time obstacle motion prediction for dynamic collision avoidance, 2019. arXiv:1811.01075.

[45] P. Long, W. Liu, J. Pan, Deep-learned collision avoidance policy for distributed multi-agent navigation, 2017. arXiv:1609.06838.

[46] P. Long, T. Fan, X. Liao, W. Liu, H. Zhang, J. Pan, Towards optimally decentralized multi-robot collision avoidance via deep reinforcement learning, 2018. arXiv:1709.10082.

[47] D. V. Lu, D. Hershberger, W. D. Smart, Layered costmaps for context-sensitive navigation, in: 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014, pp. 709–715. doi:10.1109/IROS.2014.6942636.

[48] ROS contributors, nav2: Navigation2 stack for ros2, ROS2 Ecosystem, 2024. URL: https://navigation.ros.org/#, accessed: 2024-01-12.

[49] ROS contributors, Ros (robot operating system), 2024. URL: https://www.ros.org, accessed: 2024-01-12.

[50] ROS contributors, Collision monitor node, ROS2 Ecosystem, 2024. URL: https://https://navigation.ros.org/configuration/packages/collision_monitor/configuring-collision-monitor-node.html, accessed: 2024-01-12.

[51] J. van den Berg, M. Lin, D. Manocha, Reciprocal velocity obstacles for real-time multi-agent navigation, in: 2008 IEEE International Conference on Robotics and Automation, 2008, pp. 1928–1935. doi:10.1109/ROBOT.2008.4543489.

[52] J. van den Berg, S. Guy, M. Lin, D. Manocha, Reciprocal n-Body Collision Avoidance, volume 70, 2011, pp. 3–19. doi:10.1007/978-3-642-19457-3_1.

[53] N. Vinukonda, K. N. Shah, Dynamic obstacle avoidance for assistive autonomy using orca, 2024. URL: https://nagarjunvinukonda.github.io/files/DR_Obstacle_Report.pdf, accessed: 2024-01-12.

[54] H. Wen, J. Yang, Research on a costmap that can change the turning path of mobile robot, Journal of Physics: Conference Series 2005 (2021) 012095. URL: https://dx.doi.org/10.1088/1742-6596/2005/1/012095. doi:10.1088/1742-6596/2005/1/012095.