Assistive Robot for Safe Navigation and Object Detection for Visually Impaired Individuals with Audio Feedback

Tarfah Al-Maghlouth, Sama Alzahrani, Eesa Bazarwala, Yixuan Zhu

***Abstract*—This project focuses on designing an assistive robotic system to help visually impaired individuals navigate safely by providing real-time audio feedback. The system integrates four major modules: Simultaneous Localization and Mapping (SLAM), A star (A*) search pathfinding, You Only Look Once (YOLO) object detection, and Text-to-Speech (TTS), all coordinated through a Python control architecture. SLAM continuously generates maps of the environment and estimates the robot's location, while the A* algorithm computes optimal paths. YOLO performs real-time object detection, and the TTS module delivers concise voice alerts. Additionally, in order to avoid collisions, an object avoidance method is incorporated using distance sensor data. The system was tested in Webots simulation using Hemission robot, with performance evaluated through trials.**

***Index Terms*—Assistive Robotic, visual impairment, SLAM, YOLO object detection, A* search pathfinding, Text-to-Speech, Object avoidance.**

I. Introduction

ARTIFICIAL intelligence and robotics have opened new possibilities for assistive technologies, particularly for individuals with visual impairments. Navigation is a daily challenge for visually impaired, and traditional aids such as white canes or guide dogs lack ability to detect and describe objects at a distance. Assistive robots can enhance spatial awareness through intelligent sensing and real time communication [1]. This project proposes a multimodal assistive navigation system integrating SLAM, A* path planning, YOLO object detection, and TTS. SLAM simultaneously constructs a map and estimates the robot's pose [2]. The A* algorithm leverage this map to compute efficient paths [5], while YOLO detects indoor obstacles in real time. These detections are then translated into auditory messages via the TTS module, allowing users to understand and traverse in a safe manner. To further enhance safety, the system also integrates a LiDAR-based obstacle avoidance module, enabling the robot to respond autonomously to collisions. The system is deployed in a simulated indoor environment using Webots with the **Hemisson** robot navigating an adapted apartment scenario. This paper explores whether the combination of these modules can provide effective, real-time navigation support for visually impaired users in structured indoor spaces.

The link to the code repository is provided here: https://github.com/Samm-006/Robotics-Group39.git.

II. Related Work

Many researchers have developed assistive robots to help visually impaired people move safely by combining mapping, path planning, object detection, and audio feedback. SLAM is widely used in mobile robots to build maps while tracking their position. Studies like [2] and [15] show how SLAM works well in unknown environments, especially when using grid maps. A* is one of the most popular algorithms for finding paths in robot navigation. Research by Raza et al. [7] and Xiang et al. [5] shows how A* can find the best route on a grid. These makes A* a solid choice for assistive robots that need to plan safe paths quickly. For object avoidance, the YOLO system is known for being both fast and accurate. It runs in real time and works well with cameras in indoor settings. YOLOv8n [14] is a lighter version, which makes it ideal for this kind of system. Past work like [13] has shown how YOLO helps in guiding visually impaired users by detecting obstacles early. Audio feedback is key for visually impaired users. Yao et al. [8] and Messi et al. [9] explains how sound based systems help users stay aware of their surroundings. Other systems like [10] also use sound to warn users about nearby obstacles using stereo vision and speech. This project brings together these ideas, SLAM, A*, YOLO, and TTS into one working system and tests it in a simulated indoor worlds using the Hemisson robot.

III. SLAM [Tarfah Al-maghlouth]

For indoor navigation systems designed to assist visually impaired individuals, it is essential for the robot to construct a reliable and up to date map of its environment while simultaneously determining its location within it. This functionality is provided by Simultaneous Localization and Mapping (SLAM), a foundational capability in mobile robotics [15]. Slam eliminates the need for preloaded maps, which is especially useful in indoor spaces where layouts may change.

A. Slam Method Selection

Our implementation uses a grid based SLAM approach combined with odometry and infrared (IR) proximity sensing. The environment is divided into a 2D occupancy grid, where each cell represents either free space or an obstacle. The grid is updated incrementally as the robot moves, providing real time map suitable for assistive navigation. We selected this approach for its computational efficiency and suitability for structured indoor environment [16] [17]. Unlike visual SLAM methods(e.g., ORB SLAM), our approach doesn't require high performance processors [18].

B. Tracking the Robot's Position (Odometry)

The Hemission robot uses differential drive odometry to estimate its position. Encoders on the left and right wheels

provide the distance each wheel travels. These measurements are used to compute the robot's displacement and rotation. Specifically, the translational movement d_C and change in orientation $d\theta$ are calculated using:

$$d_c = \frac{\Delta L + \Delta R}{2}, \quad d\theta = \frac{\Delta R - \Delta L}{L}$$

where $DeltaL$ and $DeltaR$ are changes in left and right wheel encoder readings, and L is the distance between the wheels. The robot's pose (x, y, θ) is then updated using:

$$x \mathrel{+}= d_C \cdot \cos\left(\theta + \frac{d\theta}{2}\right), \quad y \mathrel{+}= d_C \cdot \sin\left(\theta + \frac{d\theta}{2}\right)$$

This update is applied in each control cycle to maintain an estimate of the robot's position grid.

C. Grid Mapping

To improve mapping accuracy, IR sensors positioned around the robot provide proximity data. These readings, combined with the robot's pose, are used to determine the location of the obstacle relative to the robot. Obstacle cells in the occupancy grid are incrementally increased in confidence(e.g., +0.2), while free cells decreased (e.g., -0.05) and later binarized for path planning.

D. Slam integration

The SLAM process runs continuously until a threshold of explored cells is met, at which point it transitions to the A* path finding module. This threshold ensures that the robot has a usable information of the environment before attempting to plan a route.

SLAM is the first step in the robot's controller. The generated occupancy grid feeds directly into A* pathfinding algorithm, enabling a safe route to the destination.

Our grid based SLAM implementation is intentionally lightweight and real time making it practical for systems and gives accurate enough maps to help the robot avoid obstacles and guide the user safely.

E. SLAM Algorithm Pseudocode

Algorithm 1 Grid Based SLAM

Initialize grid $G[N][N] \leftarrow 0$, pose $(x, y, \theta) \leftarrow (0, 0, 0)$
while robot is active **do**
  Read left and right encoders: ΔL, ΔR
  Compute d_C and $d\theta$ using odometry equations
  update pose (x, y, θ)
  Update current cell in grid: $G[i][j] \mathrel{-}= 0.05$
  for each IR sensor s_i **do**
    Convert s_i to global coordinates (x', y')
    Map (x', y') to grid cell $G[i][j]$
    Update: $G[i][j] \mathrel{+}= 0.2$
  end for
  Clip G values between $[-5.0, 5.0]$
  Binarize G for A* if threshold is reached
end while

IV. A* Search Pathfinding [Eesa Bazarwala]

The main component of an assisted navigation system is path planning, which predicts safe and effective pathways for the robot to follow. According to research, reference path planning for blind and visually impaired persons (BVIP) must optimise distance, safety and support for BVIP mobility [9]. Considering its optimality and efficiency for static, grid-based maps created via SLAM, the A* algorithm is one of the most popular robotic navigation algorithms in use for path planning [5]. Using Webots, there is evidence that the algorithm is compatible with finding the optimal path within a simulated robot and environment [6].

Many papers use A* search for different robotic applications and adapt the algorithm according to their practical needs. For example, this paper [3] uses Dynamic A* Lite search (D* Lite) to find an optimal path in a real-world, dynamic environment using a Pepper robot for facial recognition for lab security. Another example is in this paper [4] where they apply optimisation techniques such as a a bidirectional A* search and curve smoothing for a better path planning and lower computational algorithm. For this project, the original A* algorithm was used and the strengths and weaknesses was explored for the simulated robot.

A. Algorithm

The A* algorithm used is obtained from this paper [7]. It contains the pseudocode for the algorithm implementation, which was adapted for this task. The algorithm works as follows: it first takes an input of the static environment grid with the robot position and the goal it would like to reach. Each cell in the grid was represented as a node in a graph. It uses an evaluation function f(n) to determine which node it would expand next. F(n) is made up of two parts, g(n) and h(n), where f(n) = g(n) + h(n). H(n) is the heuristic function which is used to estimate the distance from a certain node to the goal. As recommended by this paper [7], Manhattan or Euclidean distance is normally used, so Manhattan distance was implemented for the code, as diagonals were not included in the movement. G(n) is the cost to get from the start to the current node. For every step, the cost was equal to 1, so g(n) will represent the current, shortest number of steps it takes to reach the node. For the current node, it expands all possible neighbour nodes and calculates the f(n) value of each node. If the f(n) value already exists for that node, it updates it with the cheapest f(n) value. The neighbour nodes are one position up, down, left and right from the current cell on the grid. According to this paper [11], a priority queue was used to keep track of the cheapest f(n) values with their corresponding nodes for the most efficient algorithm. This project used a heapdict, which is a priority queue and dictionary, because it allows editing of the priority queue. The priority queue chooses the node with the cheapest f(n) value to expand next, regardless of whether it was just added or was already in the queue. The algorithm keeps track of each node and the node it was expanded from using a dictionary, so the algorithm can create a final path at the end. Once the algorithm chooses the

end goal as the current node, the algorithm terminates, and a final, optimal path is created. If the priority queue is empty without it reaching the end, i.e there are no more possible nodes to check on the grid, the algorithm terminates, and no path is possible. To test whether the algorithm works, 3 test environmental grids were created. One with multiple paths available, one with a single path available and one with no paths available. These are used to test every single use case. With the test grids, each cell is labelled as either 1 or 0, where 1 means the cell is free and 0 means the cell is blocked. To test these, uncomment the test grid, start and end positions in the code. Optionally, the SLAM threshold could also be changed to 0 to skip SLAM.

B. Implementation

Once the algorithm is created, it needs to be implemented to the robot and tested. The start node was the SLAM's robot pose output, translated to the position on the grid. The end node represents the position that the visually impaired person would like to reach. Since this is a simulated world, to implement the end node for testing, a random node of the grid which is accessible is chosen; i.e where the grid is 1. Once the start and end node was established, using the grid from the SLAM's output, the A* algorithm calculated the path and moved the robot to the end node via the optimal path.

The movement works as follows: it first turns towards the orientation of the direction the robot would want to go. Once the average encoder values reach the angle, the robot moves forward. It moves forward a certain distance to move from one cell in the grid to the next until the average encoder values reach the angle that corresponds to the distance. It alternates between turning and moving forward until the end goal has been reached. In the code, this process was represented as a finite state machine where the first state would be "SLAM". Once SLAM finishes, the state changes to "A star". A* calculates the optimal path and the states alternate between "Turn" and "Forward" to move the robot. Once the end node has been reached, the final state is "Idle", where it terminates. The hypotheses to test the algorithm works is that the A* search works perfectly with all 3 test environments as stated above, and the robot is able to follow the path and move where required. It should also be tested 5 times with SLAM to see if the algorithm can be integrated correctly.

V. YOLO Object Detection[Yixuan Zhu]

Real-time object detection is essential for assistive navigation systems supporting visually impaired users. This part employs the You Only Look Once (YOLO) framework to achieve fast and accurate obstacle recognition within the robot's camera field of view. As a single-stage detector, YOLO performs localization and classification in one forward pass, enabling rapid perception suited to dynamic indoor environments. [12]

To meet the real-time requirements of robot navigation, we conducted research and testing and ultimately chose the lightweight YOLOv8n (Nano) model. [14] Its low computational cost and stable detection accuracy allow smooth deployment in the Webots simulation and Python control architecture. YOLOv8n detects common indoor obstacles—such as tables, chairs, and wall edges—and its outputs are integrated with SLAM and the TTS module to generate spatially meaningful voice guidance (e.g., "Table detected 1 meter ahead").

A. Algorithm

For this assistive robot, real-time performance is critical to ensure timely audio feedback for the visually impaired user.Previous studies have demonstrated the effectiveness of YOLO-based systems for blind navigation tasks due to their high inference speed [13].We utilized the YOLOv8n model from the ultralytics library. The "Nano" version was selected because it balances detection accuracy with computational efficiency, making it suitable for robotic simulation environments where system resources are shared with SLAM and path planning modules.

The system captures visual data using the Webots camera device. Since Webots cameras output raw image data in BGRA format (Blue, Green, Red, Alpha), a preprocessing step is required to convert this into the BGR format compatible with the YOLO model.

B. Implementation

The implementation relies on the opencv-python library for image matrix manipulation. A critical component of our implementation is the integration logic between visual data and the distance sensor. The YOLO model is trained on distinct objects (e.g., persons, chairs, potted plants). However, in an indoor environment, "walls" are ubiquitous obstacles but are rarely classified as distinct objects by standard pre-trained models.To address this, we implemented a heuristic fusion logic: if the distance sensor detects an obstacle within 0.5 meters ($D < 0.5$), but the YOLO model returns an empty set of detected objects ($S_{obj} = \emptyset$), the system infers that the obstacle is a "Wall". This classification is then passed to the TTS module to warn the user.

C. YOLO Integration in the System

In this system, YOLOv8n performs real-time visual detection in each control cycle, converting the BGRA images from the Webots camera into BGR images before streaming inference, and outputting the target category, confidence score, and bounding box. The system filters targets with a confidence score higher than 0.5 and sends them to TTS for broadcast, while using historical records to avoid duplicate prompts, thus improving the efficiency and stability of voice feedback.

Furthermore, YOLO's detection results are combined with information from distance sensors and LiDAR to compensate for the limitations of the camera's field of view. When an object is too close to a wall and YOLO fails to detect it, the system automatically identifies it as a wall and triggers a corresponding voice prompt. The images detected by YOLO are simultaneously displayed via OpenCV, providing visual

debugging support and aiding in the analysis of target detection performance and the robot's behavior in the environment.

Overall, YOLOv8n provides real-time and stable visual perception throughout the robot's cyclic control. The detection results are used for voice prompts, obstacle recognition, and obstacle avoidance assistance, making it a crucial component of the entire multimodal navigation system.

VI. Text to speech [Sama Alzahrani]

Visually impaired users rely mainly on audio signals to understand their surroundings. Traditional aids, such as white canes or guide dogs, do not identify objects or positions. To improve awareness of the environment, an assistive system integrates computer vision with speech output. Recent studies have highlighted that audio feedback is an effective and widely used method of delivering navigational information to people with impaired vision. According to research [8], assistive systems for the visually impaired employ three type of feedback: visual, audio and haptic, with audio being the most practical for impaired users.

This project explored the possibility of improving the safety of audio feedback among visually impaired users. It is concerned with aligning the text-to-speech results with real-time data processed by SLAM, A-star, and Yolo software. The suggested system converts identified obstacles and routing messages into concise voice alerts that are crafted to be easily understandable and instant. This approach is consistent with most assistive robotics designs, such as audio-augmented navigation systems that use computer vision and BIM [9] and stereo-vision-based obstacle detection with spoken warning [10].

The article is a critical review of the known best practice in assistive audio cues and also it experiments the accuracy of user training in the Webots simulation platform. The overall purpose is to provide guidance in time that will maximise the user confidence and minimise the risks of navigation. The study also examines the effect of prompt timing, comprehensibility and design on fluidity and efficiency of movement in dynamic indoor environments. The results show that properly designed auditory feedback makes the navigation and the overall functioning of the visually impaired individuals easier and more efficient.

A. Implementation

For real time speech output, the message is processed using the `pyttsx3` TTS engine. The feedback is based on the output received from LiDAR, YOLO, and the distance sensor. When an object with a confidence level `>0.5` is identified by YOLO, the system forms a short voice message by joining the detected labels and converting them to text-to-speech, such as "Potted plant ahead, avoid hitting it". To prevent overlapping and repetitive warnings, the system stores the last announced object and only triggers speech when a new object is detected. This ensures that the robot completes one sentence before beginning the next and avoids repetition that may be intimidating. Moreover, based on its movement, the robot also provides spoken alerts. For instance, it displays "Moving forward" and "Turning now", or "Wall ahead, avoid hitting it" when the risk is detected during avoidance. This system helps visually impaired individuals understand what is in front of the robot and how it is responding by combining object-based warnings with movement-based feedback to deliver clear speech.

VII. Object Avoidance [Sama Alzahrani]

LiDAR-based distance sensing and a finite-state control system were combined to achieve object avoidance. This helps the robot identify surrounding objects that could be outside the camera's field of view. Therefore, LiDAR was implemented to ensure safety and avoid collisions, which continually scans the area directly in front of it using a 360° LiDAR Sensor (LDS-01). This is a 2D laser scanner capable of sensing 360 degrees that collects a set of data around the robot to use for SLAM and Navigation. Through the integration of LiDAR-based proximity sensing and visual recognition, the robot can recognise obstacles ahead of time and prompt action to avoid collision.

Object avoidance operates alongside SLAM. SLAM continually updates the robot's position and occupancy grid as object avoidance responds to nearby obstacles. This implies that odometry and sensor updates continue to be used even in the REVERSE and TURN status. Due to this, the robot responds to unknown obstacles while maintaining the map consistency, that is crucial for navigation in a cluttered indoor environment.

A. Implementation

For avoidance, only the front sector of the LiDAR scan is retrieved at each simulation step. The minimal distance to any object is considered using this sector, which represents the area in front of the robot.Additionally, SLAM operates alongside the object avoidance system, allowing the robot to continue building the map while avoiding obstacles using LiDAR. As the robot continues, the distance stays above a certain threshold. Therefore, if the distance between the robot and the object is less than 0.3m, the system switches to a finite state avoidance controller of three states:

1) FORWARD state: The robot moves forward at a constant speed (`CRUISE_SPEED = 6.28`). When LiDAR detects an obstacle within 0.3m, it switches to "REVERSE" state. TTS here send feedback "Moving Forward" to alert the direction of movement.
2) REVERSE state: the robot starts reversing for a certain amount of time to make space between the robot and the object detected, then switches to the TURN state.
3) TURN state: TTS alters the individual with "Turning now" to make them aware. Then returns to FORWARD if the LiDAR reading indicates a clear route.

B. Algorithm

Algorithm 2 Object Avoidance Finite State Machine

while robot is active **do**
  Read LiDAR front sector and compute min_front
  if $robot\_state =$ FORWARD **then**
    if $step\_count \bmod TTS\_INTERVAL = 0$ **then**
      Speak 'Moving forward'
    end if
    if $min\_front < OBSTACLE\_DIST$ **then**
      $robot\_state =$ REVERSE
      $state\_timer = 60$
    else
      $left\_motor = CRUISE\_SPEED$
      $right\_motor = CRUISE\_SPEED$
    end if
  end if
  if $robot\_state =$ REVERSE **then**
    if $step\_timer > 0$ **then**
      $left\_motor - 1.0$
      $right\_motor - 1.0$
      $state\_timer = state\_timer - 1$
    else
      Speak 'Turning now'
      $robot\_state =$ TURN
      $robot\_timer = 60$
      Choose random direction $d \in \{-1, +1\}$
      $current\_turn\_velocity = TURN\_SPEED \cdot d$
    end if
  end if
  if $robot\_state =$ TURN **then**
    if $step\_timer > 0$ **then**
      $left\_motor - current\_turn\_velocity$
      $right\_motor + current\_turn\_velocity$
      $state\_timer = state\_timer - 1$
    else
      if $min\_front > OBSTACLE\_DIST + 0.1$ **then**
        $robot\_state =$ FORWARD
      else
        $state\_timer = 30$ {Continue turning}
      end if
    end if
  end if
end while

VIII. Conclusion, Evaluation and Limitations

The algorithms stated above were integrated to create one working system. It worked as follows: SLAM and object avoidance map the room and localise the robot. A* calculates the optimal path using the output of SLAM. The robot moves using the optimal path where YOLO detects objects, and TTS recalls the object to the visually impaired person. This combination created a successful integrated Python controller.

The implementation of the integrated assistive navigation system using Webot in an indoor environment encountered a range of technical challenges and practical limitations due to the combined use of SLAM, A* path planning, YOLO object detection, LiDAR-based avoidance, and Text-to-Speech feedback. It is important to evaluate these algorithms to allow further improvement and development for future use and analysis.

A. Evaluation of SLAM and it's Limitations [Tarfah Almaghlouth]

We hypothesized that the grid based SLAM implementation would enable the robot to generate a usable occupancy map and allow successful A* path planning in at least 80% of indoor test runs under normal operating condition.

In testing conducted within the Webots simulator, the SLAM module reliably produces occupancy grids that captured key features of the environment, such as open space and obstacles. These maps enabled he robot o proceed with A* pathfinding and successfully navigate toward randomly selected targets in most trials. The SLAM system was responsive in real time, allowing incremental map updates and consistent integration with other modules.

Main limitation observed while testing was tied to the probabilistic nature of the occupancy grid. Since obstacles are incrementally marked with small values (e.g., +0.2) and cleared with decrements (e.g., -0.05), cells may never fully reach the obstacle threshold if the robot doesn't repeatedly observe them. As a result, some obstacles are unrepresented in the binary map, leading to possible path planning failures. Adjusting the thresholds or refining the confidence update logic could reduce this issue.

Additionally, SLAM does not include any loop closure or global correction step, meaning that long term navigation can cause map drift, especially if the robot returns to previously visited areas. This limitation is acceptable in small scale assistive indoor settings but would affect performance in larger environments.

Despite these limitations, SLAM in this system provides sufficient real time mapping and localization for path planning. It simplicity allows it to run efficiently alongside YOLO and A* algorithm without overloading the simulation. Overall, the implementation meets the practical needs of assistive indoor navigation but could benefit from improvements in sensor fusion, obstacle confidence management, and pose correction.

B. Evaluation of the A* Algorithm and it's Limitations [Eesa Bazarwala]

Evaluating the algorithm, the worst-case space complexity is O(N^2), where NxN would be the size of the grid. The worst-case time complexity is also O(N^2) as the while loop could go through every cell in the grid before reaching the goal. Evaluating the hypotheses, from the 3 test grids as stated above, the algorithm works perfectly with all 3 grids, producing the most optimal path where it's required, and the robot moves in the correct path as shown in the video. Similarly, when testing with the integrated system, once SLAM has mapped part of the room, A* can calculate the shortest path and move towards the goal given the output of SLAM. Running SLAM five times for a few minutes, it each

prints out a slightly different grid which is tested with A*. Three times it completed the path, one time it hit an object and ended somewhere else and the last time it could not find a path.

There exists limitations with the A* algorithm that was discovered during testing. Firstly, A* depends strongly on the SLAM output. If SLAM labelled a free space as blocked in the grid or vice versa, the path calculated by A* can be non-optimal or can crash into objects. Also, this can cause situations where even if a path is possible, A* can't find a path due to a node that should be free, is considered blocked. Secondly, since the algorithm only considers 4 directions, up, down, left and right, the optimal path is jagged and not smooth as it does not consider diagonals. Curve smoothing can improve this as mentioned in this paper [4]. Thirdly, during the robot movement phase, small errors can occur during the odometry, which can cause the robot to overshoot. These errors can accumulate if the path is long and the robot can end at a different place than intended.

C. Evaluation of YOLO and it's Limitations[Yixuan Zhu]

From the test cases conducted, YOLOv8n performs robustly under normal lighting conditions and produces stable bounding boxes for medium-sized indoor objects. When integrated with the full system, the detection results, combined with SLAM-based localization, allow the robot to announce obstacles at appropriate distances. Testing shows that YOLOv8n consistently detects major obstacles; however, occasional missed detections occur when objects are partially occluded or when the camera angle provides insufficient visual information.

First, there's the issue of object misclassification. If the robot encounters a non-wall obstacle (e.g., a specific trash can or other obstacle not included in the YOLO training set) and approaches it, the system will incorrectly label it as a "wall" because it cannot recognize the object category. Second, there's a near-field blind spot. When the robot approaches an object, the camera's field of view narrows. If the object occupies the entire frame, the features required for YOLO detection may be occluded, leading to pre-collision detection failure.Thirdly, due to the lightweight nature of the Nano model, its accuracy is lower than that of larger YOLO variants, and detection jitter occasionally appears when the robot moves quickly. These issues can be mitigated by integrating temporal smoothing, confidence filtering, or upgrading to a larger YOLOv8 model depending on computational capacity.

D. Evaluation of TTS and Object Avoidance and it's Limitations [Sama Alzahrani]

1) Text-To-Speech: The performance of TTS in this assistive system, which helps give impaired individuals an ongoing alertness to their surroundings, is reliable. Some challenges encountered during the testing of TTS were the overlapping and repetition of alerts when continuous objects are detected using YOLO and LiDAR readings. The `tts_cooldown` variable is used as a cooldown timer. The purpose of it is to ensure that each message is finished before the next to avoid overlapping. Moreover, the `last_detected_obj` variable saves to memory the most recent spoken message, helping minimise repetitive prompts.

Some limitations that may be considered in TTS from testing are delays or missing alerts due to restrictions. For example, while YOLO continuously detects, the cooldown is still active, resulting in the robot not making an alert on the newly detected object. This makes it too slow and repetitive. Another limitation is the limited vocabulary. Due to the environment and the robot setup, it is difficult to implement TTS to make it say how far the object is and in what direction, which offers helpful feedback for users, rather than simple phrases like "Chair ahead, avoid hitting it".

2) Object Avoidance: Object avoidance prevents collisions, which plays a crucial role in the integrated system. However, several challenges observed during testing need to be highlighted. Considering the finite state machine that relies on a strict distance threshold, and the Hemission robot to be able to avoid any detected objects that YOLO is unable to identify within a close distance. This suggests the importance of LiDAR to detect close-range objects and for the robot to retreat and reorient.

During experiments, the behaviour of object avoidance was shown to be reactive rather than predictive. This means the robot only changes state after the risk has appeared rather than predicting it early, which is less efficient and results in frequent rotation in tight spaces. This suggests the system works, but with limitations. This is a limitation because sometimes it end up turning into tight spaces that would be too small for impaired users who are physically much larger than the robot. Another limitation encountered from testing is that it chooses the turn direction randomly, which raises the probability of it doing unnecessary movement.

E. Conclusion

This project combines SLAM, A* pathfinding, YOLO object detection, Text-to-Speech (TTS) and object avoidance into a single assistive navigation system for individuals with visual impairments. The robot was able to observe its surroundings, compute an optimal path, detect and avoid obstacles, and deliver real time audio feedback. Testing in the Webots simulator confirmed that the system can safely navigate structured indoor environments under controlled conditions.

The evaluation highlighted several limitations across modules, including odometry drift affecting SLAM accuracy, inconsistent map outputs, jagged paths, occasional object misclassification by YOLO, latency in TTS delivery, and the reactive nature of the obstacle avoidance strategy. These factors influenced the overall reliability and smoothness of navigation.

Despite these challenges, the integrated systems demonstrated a functional and modular framework for assistive robotics. With further refinement such as improved sensor fusion, map correction mechanisms, and enhanced obstacle prediction the system can be made more robust and adaptable for real world deployment.

REFERENCES

[1] M. Soori, B. Arezoo, and R. Dastres, "Artificial Intelligence, Machine Learning and Deep Learning in Advanced Robotics, A Review," Cognitive Robotics, vol. 3, no. 1, pp. 54-70,2023, doi: https://doi.org/10.1016/j.cogr.2023.04.001

[2] H. Ding, B. Zhang, J. Zhou, Y. Yan, G. Tian, and B.-L. Gu, "Recent developments and applications of simultaneous localization and mapping in agriculture," Journal of Field Robotics, vol. 39, no. 6, pp. 956–983, May 2022, doi: https://doi.org/10.1002/rob.22077.

[3] D. Belanová, M. Mach, P. Sinčák and K. Yoshida, "Path Planning on Robot Based on D* Lite Algorithm," 2018 World Symposium on Digital Intelligence for Systems and Machines (DISA), Košice, Slovakia, 2018, pp. 125-130, doi: https://doi.org/10.1109/DISA.2018.8490605.

[4] C. Jiang, C. Wang and M. Wang, "Research on path planning for mobile robots based on improved A-star algorithm," 2023 IEEE 7th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 2023, pp. 723-727, doi: https://doi.org/10.1109/ITOEC57671.2023.10292045.

[5] D. Xiang, H. Lin, J. Ouyang et al., "Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot," Scientific Reports, vol. 12, art. no. 13273, 2022, doi: https://doi.org/10.1038/s41598-022-17684-0

[6] G. A. Vargas, O. G. Rubiano, R. A. Castillo, O. F. Avilés and M. F. Mauledoux, "Simulation of e-puck path planning in Webots," International Journal of Applied Engineering Research, vol. 11, no. 19, pp. 9772-9775, 2016.

[7] S. J. A. Raza, N. A. Gupta, N. Chitaliya, and G. R. Sukthankar, "Real-World Modeling of a Pathfinding Robot Using Robot Operating System (ROS)," 2018, arXiv preprint, doi: https://doi.org/10.48550/arXiv.1802.10138

[8] F. Yao, W. Zhou, and H. Hu, "A Review of Vision-Based Assistive Systems for Visually Impaired People: Technologies, Applications, and Future Directions," arXiv (Cornell University), May 2025, doi: https://doi.org/10.48550/arxiv.2505.14298

[9] L. Messi, M. Vaccarini, A. Corneli, A. Carbonari, and L. Binni, "An Audio Augmented Reality Navigation System for Blind and Visually Impaired People Integrating BIM and Computer Vision," Buildings, vol. 15, no. 18, p. 3252, Sep. 2025, doi:https://doi.org/10.3390/buildings15183252.

[10] M. Kevin, C. Mario, L. Ortiz, S. Sutter, S. Klaus, and Bacca-Cortes Bladimir, "Embedded solution to detect and classify head level objects using stereo vision for visually impaired people with audio feedback," Scientific Reports, vol. 15, no. 1, May 2025, doi:https://doi.org/10.1038/s41598-025-01529-7

[11] Chatzisavvas A, Dossis M, Dasygenis M. "Optimizing Mobile Robot Navigation Based on A-Star Algorithm for Obstacle Avoidance in Smart Agriculture," Electronics. 2024; 13(11):2057. https://doi.org/10.3390/electronics13112057

[12] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788, https://doi.org/10.1109/CVPR.2016.91.

[13] S. Davanthapuram, X. Yu and J. Saniie, "Visually Impaired Indoor Navigation using YOLO Based Object Recognition, Monocular Depth Estimation and Binaural Sounds," 2021 IEEE International Conference on Electro Information Technology (EIT), Mt. Pleasant, MI, USA, 2021, pp. 173-177, https://doi.org/10.1109/EIT51626.2021.9491913

[14] M. Yaseen, "What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector," arXiv, vol. 2408.15857, 2024, doi: https://doi.org/10.48550/arXiv.2408.15857.

[15] Qiao, J., Guo, J. and Li, Y. (2024). Simultaneous localization and mapping (SLAM)-based robot localization and navigation algorithm. Applied Water Science, 14(7). doi:https://doi.org/10.1007/s13201-024-02183-6.

[16] Wu, Z., Meng, Z., Xu, Y. and Zhao, W. (2022). A Vision-Based Approach for Autonomous Motion in Cluttered Environments. Applied Sciences, [online] 12(9), pp.4420–4420. doi:https://doi.org/10.3390/app12094420.

[17] Peng, Tao & Zhang, Dingnan & Hettiarachchi, Don Lahiru Nirmal & Loomis, John. (2020). An Evaluation of Embedded GPU Systems for Visual SLAM Algorithms. Electronic Imaging. 2020. 325-1. 10.2352/ISSN.2470-1173.2020.6.IRIACV-074.

[18] Labbé, M. and Michaud, F. (2018). RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation. Journal of Field Robotics, 36(2), pp.416–446. doi:https://doi.org/10.1002/rob.21831.