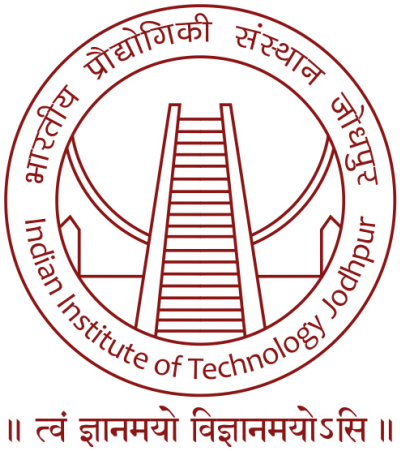
Human Following Robot using Camera and LiDAR

A B.Tech Project Report
Submitted by

Devyani Gorkar
Tejas Ingale

Under Supervision of
Dr. Riby A Boby



Date: November 23, 2023

Contents

Acknowledgment **2**

Abstract **3**

1 Introduction **3**
1.1 Objective . . . 3
1.2 Literature review . . . 3
1.2.1 Background of Autonomy . . . 3
1.2.2 Human Detection and Tracking . . . 4
1.2.3 Human Following . . . 5
1.2.4 LiDAR data processing . . . 5
1.2.5 Sensor Fusion . . . 5

2 Work Done **6**
2.1 Implementation . . . 6
2.1.1 Human following . . . 6
2.1.2 Obstacle avoidance . . . 10
2.1.3 Sensor Fusion . . . 13

3 Results and Discussion **13**
3.1 Human following robot experimental results . . . 13
3.2 Limitations and Future Scope . . . 15

4 Conclusion **15**

References **15**

Acknowledgment

We extend our sincere gratitude to Professor Riby A Boby for his invaluable guidance, unwavering support, and expertise throughout this project. His insightful feedback and encouragement have been instrumental in shaping our ideas and refining our approach.

We would like to express our appreciation to our partners, for their collaborative spirit and dedication to the project. Working together has been a rewarding experience, and their contributions have significantly enriched the overall quality of our work.

We also want to acknowledge the support and resources provided by *Mechanical Department of Indian Institute of Technology, Jodhpur*, which have played a crucial role in the successful completion of our project.

Finally, we extend our thanks to all those who have shared their knowledge and expertise in the field of robotics, computer vision, and LiDAR technology. Your insights have been instrumental in shaping the development of our Human Following Robot using Camera and LiDAR.

This project has been a challenging yet fulfilling journey, and we are grateful to everyone who has been a part of it.

Abstract

The goal of this project is to develop robots that can track humans autonomously in human-centered settings, using a combination of cameras and LiDAR sensors. The algorithm processes the data from both sensors to generate a real-time representation of the environment, which includes the depth from LiDAR and appearance of human targets from camera. The algorithm enables the robot to follow humans reliably and dynamically. The project has potential applications in various domains, such as warehouse automation, personal robotics, and healthcare, demonstrating improvement in human-robot interaction.

1 Introduction

1.1 Objective

Our project aims to develop a **computer vision** algorithm that combines two sensor modalities using the sensor fusion technique to make a mobile robot follow a human instructor without hitting any obstacles in its proximity. Assuming the robot works in fully automated mode until given a gesture, the gesture shifts the robot from fully automated mode to semi-automatic mode, where the robot's task is to follow the human being while avoiding obstacles.

1.2 Literature review

Human-following robots are widely used in industries and for research purposes. Researchers have used different approaches to achieve human following. For human detection, feature-based analysis [1] - [3] and skeleton detection strategies [4], are very often adopted by the designer but the accuracy can be affected by the changing environmental conditions. Considering the influence of environmental factors, several environmental sensors are employed for human detection, such as ultrasonic sensors, 2D Lidar, RGB cameras, and RGB-D cameras. Ultrasonic sensors is relatively inexpensive, but it is susceptible to interference for poor angular resolution. Afghani et al. [5] used ultrasonic sensors to detect environmental obstacle information, and then segment the sensing range of the IR sensor to determine whether to avoid obstacle or follow the target.[6] used 2D Lidar to detect human waste characteristics and environmental information. Cameras can provide color images to fully represent the features of the object. RGB-D cameras can provide both RGB images and depth images.

1.2.1 Background of Autonomy

Autonomous systems refer to systems capable of performing tasks or making decisions without direct human intervention. These systems can operate independently or with minimal human guidance, utilizing sensors, algorithms, and control mechanisms to perceive their environment and respond accordingly.

The levels of autonomy in autonomous systems are often categorized into different stages, ranging from manual control to fully automatic operation. In the context of robotics, the levels of autonomy are commonly described as follows:

1. Manual Control: In this level, humans have direct control over the system's actions. Operators or users are responsible for making decisions and providing instructions. The system lacks the ability to make decisions on its own and relies entirely on human input.

2. Shared Autonomy: Shared autonomy involves a collaboration between humans and machines, where both contribute to the decision-making process. The system can perform certain tasks

autonomously, but human intervention is still required for more complex decisions or in unpredictable situations. Human operators may have the ability to override or guide the system's actions.

3. Fully Automatic: At this level, the system is capable of operating independently without human intervention. It can make decisions, adapt to changing conditions, and execute tasks without continuous oversight. The system relies on its own sensors, algorithms, and programming to navigate and respond to the environment.

Our work falls within the spectrum of shared and complete autonomous systems. By fusing Lidar and camera information, we aim to enhance the robot's perception and decision-making capabilities, enabling it to autonomously track and follow human subjects.

1.2.2 Human Detection and Tracking

Human detection and tracking using computer vision have become pivotal in a myriad of applications, ranging from surveillance and security to human-computer interaction and autonomous systems.

1. Human Detection Techniques:

   - Haar-like Features and Cascade Classifiers: Viola-Jones algorithm, employing Haar-like features and cascade classifiers, was an early and influential approach for real-time human detection. Its simplicity and efficiency paved the way for subsequent developments.
   - Histogram of Oriented Gradients (HOG): The HOG descriptor captures local intensity gradients, forming a representation robust to changes in illumination. Coupled with Support Vector Machines (SVM), HOG has demonstrated effectiveness in detecting humans in images and videos.
   - Deep Learning Approaches: Convolutional Neural Networks (CNNs) and their variants, such as Region-based CNNs (R-CNN), Fast R-CNN, and Faster R-CNN, have revolutionized human detection. Models like You Only Look Once (YOLO) and Single Shot MultiBox Detector (SSD) provide real-time detection by framing it as an object detection problem.

2. Human Tracking Techniques:

   - Kalman Filters: Kalman Filters are widely used for human tracking due to their simplicity and efficiency. They predict the next state of the object and update it based on the observed measurements, making them suitable for real-time applications.
   - Particle Filters: Particle Filters are employed for non-linear and non-Gaussian tracking problems. They represent the state of an object with a set of particles, updating and resampling them based on measurements to estimate the object's trajectory.
   - DeepSORT: DeepSORT (Deep Learning for Object Tracking in Real Time) integrates deep appearance features with traditional methods, enhancing the tracking performance, especially in scenarios with occlusions and crowded environments.
   - OpenCV Trackers: OpenCV, a widely used computer vision library, provides various tracking algorithms. Notable trackers include the MedianFlow, MOSSE (Minimum Output Sum of Squared Error), and CSRT (Discriminative Correlation Filter with Channel and Spatial Reliability). These trackers leverage different strategies, such as correlation filtering and spatial reliability modeling, to achieve robust and real-time human tracking.

1.2.3 Human Following

1. Traditional Computer Vision Approaches: Early efforts in human detection primarily relied on traditional computer vision techniques. These approaches often involved extracting handcrafted features, such as edges, colors, and textures, from images. Methods like Histogram of Oriented Gradients (HOG) and Haar-like features have been extensively used. While effective to some extent, these methods are limited in handling variations in pose, illumination, and occlusion.

2. Deep Learning based methods: The advent of deep learning has revolutionized human detection using cameras. Convolutional Neural Networks (CNNs) have shown remarkable success in automatically learning hierarchical features from raw image data. Region-based CNNs (R-CNN), You Only Look Once (YOLO), and Single Shot MultiBox Detector (SSD) are among the popular architectures employed for human detection. These deep learning models excel in capturing complex patterns and contextual information, leading to improved accuracy and robustness.

3. Transfer Learning and Pre-trained Models: Transfer learning has gained prominence in human detection tasks using cameras. Pre-trained models on large datasets, such as ImageNet, are fine-tuned for specific human detection tasks. This approach allows leveraging the knowledge gained from general visual features to enhance performance on human-specific features, especially in scenarios with limited labeled data.

1.2.4 LiDAR data processing

LiDAR (Light Detection and Ranging) technology has gained significant traction in various fields, particularly in obstacle avoidance systems for autonomous vehicles and robotics. LiDAR data processing plays a crucial role in extracting meaningful information from the raw point cloud data captured by LiDAR sensors.

LiDAR sensors emit laser beams and measure the time it takes for the beams to return after hitting objects, creating a three-dimensional point cloud. The raw point cloud data generated by LiDAR sensors is typically massive, requiring sophisticated processing techniques to extract useful information. Several key steps are involved in LiDAR data processing:

1. Data Filtering and Cleaning: Raw LiDAR data often contains outliers, noise, and non-ground points. Filtering and cleaning algorithms are applied to remove unwanted points and enhance the accuracy of subsequent analyses.

2. Point Cloud Segmentation: Segmentation involves grouping points into meaningful clusters based on characteristics such as distance, intensity, or reflectance. This step is crucial for identifying objects in the environment.

3. Feature Extraction: Extracting relevant features from the point cloud, such as object boundaries, shapes, and orientations, is essential for further analysis. Feature-extraction algorithms play a key role in identifying obstacles.

4. Object Recognition and Classification: Advanced algorithms are employed to recognize and classify objects within the point cloud. This step is critical for distinguishing between obstacles, road infrastructure, and other elements in the environment.

1.2.5 Sensor Fusion

Sensor fusion aims to combine the strengths of diverse sensors to compensate for individual limitations and improve overall system performance. The fusion of data from sensors with different modalities, such as cameras, LiDAR, radar, inertial sensors, and more, has gained prominence for

its ability to provide a more complete and accurate representation of the environment.

Multi-Modal Sensor Fusion Approaches:

- Data-Level Fusion: Data-level fusion involves the direct combination of raw sensor data. Algorithms such as Kalman filters, Particle filters, and Bayesian methods are commonly employed to merge information from different sensors, ensuring a coherent and accurate representation of the environment.
- Feature-Level Fusion: Feature-level fusion focuses on extracting relevant features from each sensor modality and combining them to enhance the overall understanding of the scene. This approach often involves the use of advanced computer vision techniques, such as object recognition and image processing.
- Decision-Level Fusion: Decision-level fusion combines the decisions or outputs from individual sensors. Techniques such as fuzzy logic, neural networks, and rule-based systems are applied to reconcile conflicting information and make final decisions based on the collective sensor inputs.

2 Work Done

2.1 Implementation

There are two main components of this project 1) Human following 2) Obstacle avoidance. These two components are integrated together to for accurate and safe human following. The two components are discussed in detail below:

2.1.1 Human following

Human following is the primary objective of the project and it is done using a RGBD camera.the system employs advanced computer vision algorithms to detect and track humans in the robot's surroundings.Depth information aids in accurate identification and localization of the human target, allowing for a reliable tracking mechanism.

The human is detected in the surrounding using object detection model. After evaluating multiple object detection algorithms including SSD Lite, SSD, FasterRCNN, and YOLO, we chose YOLO as our final model. YOLO excelled in real-time processing speed, offered a unified framework, demonstrated effectiveness across scales, and maintained a favorable balance between speed and accuracy—ideal for our specific use case. The following images display the results of various object detection models.

Figure 1: Human detection using different DL models

YOLO divides an image into a grid and predicts bounding boxes and class probabilities for each grid cell simultaneously, making it efficient for real-time applications. The model is trained on the COCO dataset and accurately detects humans in the simulation. Along with human detection, we have also added human tracking to continuously monitor human motion. Tracking allows for the analysis of spatial and temporal human behavior in consecutive frames. We have used a robust tracking algorithm CSRT (Continuous Adaptive Mean-Shift with Reinforcement Tracking). CSRT uses a correlation filter to compare the target model with the surrounding image in subsequent frames.CSRT incorporates reinforcement learning techniques to adapt the model during tracking. The learning process helps the tracker to adapt to changes in the appearance, scale, and orientation of the target over time.

(a) Person detection using YOLO

(b) Depth image of human

Figure 2: RGB and Depth Image

The object detection YOLO v3 algorithm and CSRT algorithm accurately calculate the bounding box coordinates of the human. The centroid of this bounding box is used to calculate the distance between the robot and the human. Intelrealsense R200 is a stereovision camera and utilizes two cameras for depth estimation. Given the pixel coordinates (u, v) of the bounding box centroid, the associated depth value (z) is estimated using the obtained depth image captured by the camera.

The camera displays the 2D image in the form of pixels. The relationship between the 3D world coordinates (X, Y, Z) and the 2D image coordinates (u, v) is expressed through the pinhole camera model. In the context of the pinhole camera model and 3D-to-2D projection, the camera is characterized by intrinsic parameters, extrinsic parameters, and a projection matrix.

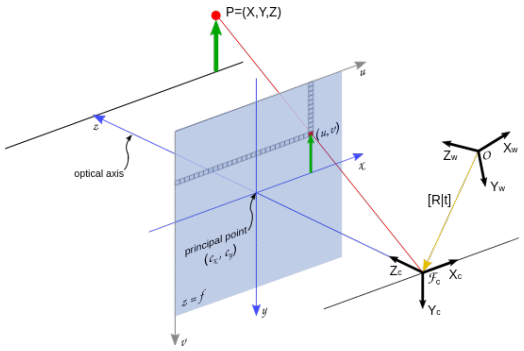


Figure 3: Pin-hole camera model

1. **Intrinsic Parameters**:
Intrinsic parameters are internal properties of the camera that define its optical characteristics. They include: Focal Length (f), Principal Point ((c_x, c_y)), Camera Calibration Matrix (K). The image coordinates can be calculated from the following equations

$$u = f_x \cdot \frac{X}{Z} + c_x$$
$$v = f_y \cdot \frac{Y}{Z} + c_y$$

$$K = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

For our camera, the camera calibration matrix was:

$$K = \begin{bmatrix} 1206.889 & 0.0 & 960.5 \\ 0 & 1206.889 & 540.5 \\ 0.0 & 0.0 & 1.0 \end{bmatrix}$$

2. **Extrinsic Parameters** Extrinsic parameters define the camera's pose in the 3D world. They include Rotation Matrix (R) and Translation Vector (T):

$$M = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$$

3. **Projection Matrix** The projection matrix (P) combines both intrinsic and extrinsic parameters. It is a 3x4 matrix defined as:

$$P = K \cdot M$$

where K is the camera calibration matrix, and M is the transformation matrix.

The projection matrix relates 3D world coordinates (X, Y, Z) to homogeneous coordinates in the image plane (u, v, w):

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = P \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

. The projection matrix for our camera was:

$$P = \begin{bmatrix} 1206.889 & 0.0 & 960.5 & -84.482 \\ 0 & 1206.889 & 540.5 & 0.0 \\ 0.0 & 0.0 & 1.0 & 0.0 \end{bmatrix}$$

These parameters can be used to estimate the position of the human in 3D world coordinate frame from the bounding box centroid coordinates in image plane.

$$X = Z \cdot \frac{u - c_x}{f_x}$$

$$Y = Z \cdot \frac{v - c_y}{f_y}$$

In our human-following mechanism, we've implemented a straightforward yet effective strategy. The camera image is partitioned by a centerline with two margins. If the human is close to the centerline, the robot moves straight, adjusting its linear velocity based on the distance to the human. When the human is positioned to the left or right of the centerline, the robot adapts its angular velocity proportionally to the distance between the centerline and the human. This approach ensures responsive and adaptive navigation depending on the human's relative position to the robot's field of view. The pseudo code for the human following algorithm is:

Algorithm 1 Human Detection and Tracking Algorithm

```
class YoloTracker
bool tracking = False
if not tracking then
    Object detection using Yolov3
else
    Object Tracking using CSRT
    Update velocity based on tracking information
    Publish velocity command
end if
```

Once the spatial coordinates of the human are computed, they are used to compute the angular and linear velocities of the human.

1. Linear velocity: Let z be the distance between the human and robot. The linear velocity of the robot is determined by

$$\mathrm{V} = \mathrm{linear\_factor} \cdot \tanh(z - \mathrm{safe\_distance})$$

2. Angular velocity: To compute angular velocity, we devide the camera frame into two parts by a line. This line is our centerline. Now, we compare X coordinates of the bounding box with the centerline and compute the angular velocity as shown in the pseudo-code. Let the minimum distance of the bounding box from the centreline be m. Then angular velocity will be

$$\omega = \mathrm{angular\_factor} \cdot \tanh(m)$$

The reason for using *the tanh* function is to overcome the inertia effects of the robot when the distance between the human and the robot reduces suddenly.

Algorithm 2 Human Following Algorithm

Initialize parameters
$centerline \leftarrow$ image_width/2
$margin \leftarrow$ some_value
while human_tracked **do**
    $human\_pos \leftarrow$ track_human()
    if human right of centerline **then**
        $linear\_velocity \leftarrow 0$
        $angular\_velocity \leftarrow angular\_factor \times tanh(human\_pos - centerline)$
    else if human left of centerline **then**
        $linear\_velocity \leftarrow 0$
        $angular\_velocity \leftarrow angular\_factor \times tanh(human\_pos - centerline)$
    else
        $linear\_velocity \leftarrow linear\_factor \times tanh(depth - safe\_distance)$
        $angular\_velocity \leftarrow 0$
    end if
    move_robot($linear\_velocity, angular\_velocity$)
end while

2.1.2 Obstacle avoidance

To facilitate obstacle avoidance, our approach incorporates an innovative Artificial Potential Field (APF) based planner seamlessly integrated with Proportional-Derivative (PID) control. The objective of this integration is to dynamically compute the linear and angular velocities of the robot in response to the environmental conditions encountered during navigation. This ensures an adaptive and responsive trajectory planning mechanism, where the robot is steered away from potential collisions while maintaining progress toward its destination.

The main principle of APF lies in modeling a robot's navigation as the movement of a particle through a potential field. This field consists of attractive forces towards the goal and repulsive forces around obstacles. The robot moves by following the steepest descent of this potential field, resulting in paths that avoid obstacles and converge toward the goal. We used the following equations to calculate the attractive and repulsive forces.

$$\mathbf{F}_{\text{att}} = k_{\text{att}} \cdot \frac{\text{goal} - \text{robot\_position}}{\|\text{goal} - \text{robot\_position}\|}$$

$$\mathbf{F}_{\text{rep}} = k_{\text{rep}} \cdot \frac{\text{obstacle} - \text{robot\_position}}{\|\text{obstacle} - \text{robot\_position}\|^3}$$

$$\mathbf{F}_{\text{net}} = \mathbf{F}_{\text{att}} + \mathbf{F}_{\text{rep}}$$

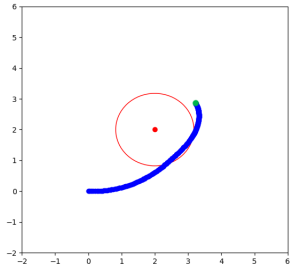
$$\mathbf{F}_{\text{net}} = F_{\text{netx}}\hat{i} + F_{\text{nety}}\hat{j}$$

$$\text{desired\_yaw} = \arctan\left(\frac{F_{\text{nety}}}{F_{\text{netx}}}\right)$$

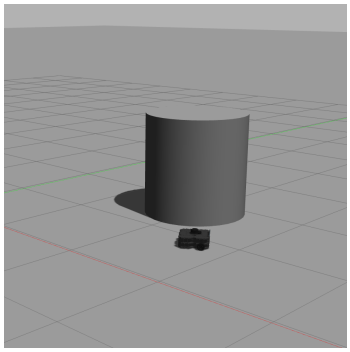
Here k_{att} and k_{rep} are positive constants determining the strength of attractive and repulsive forces. The repulsive forces only act when the robot is near the obstacle, called the region of influence of the obstacle. This prevents the trajectory of the robot from getting affected by distant obstacles. The desired yaw angle computes at every instant the desired yaw angle to reach the goal location.

The traditional APF algorithm requires a fixed and known obstacle position before generating a trajectory for the robot. To address this drawback, we have devised a novel technique to estimate the obstacle position and its region of influence in real time using the LIDAR sensor.

The LIDAR gives a range of scan values, and these values are converted to coordinates with respect to the LIDAR frame. To define an obstacle's influence region, we calculate center coordinates by computing the mean of these data points. The center coordinate is the estimated location of the obstacle. The region's radius is then established as the maximum distance from the center, further enhanced by an offset value (0.7) to ensure comprehensive coverage. When the robot enters this circular region, the repulsive force is activated. This dynamic approach enhances the robot's trajectory planning capabilities, making it more versatile and responsive in unpredictable environments. The computed center coordinates are converted from a LIDAR frame to a world coordinate frame to estimate the position of the obstacle in the environment.



(a) Trajectory followed by robot



(b) Robot and Obstacle

Figure 4: Robot avoiding obstacles

Algorithm 3 Robot Following Human with Lidar Obstacle Avoidance

```
while ¬rospy.is_shutdown() do
    Lidar Scanning:
    lidar_points = scan_world()
    Check for Nearby Obstacles:
    if distance_to_nearest_point(lidar_points) < lidar_max_distance then
        obstacle_center, obstacle_radius = estimate_obstacle_region(lidar_points)
        if distance_between_points(robot_location, obstacle_center) < obstacle_region_radius then
            human_goal = human_lastlocation
            obstacle_goal = obstacle_center
            pid_control(robot_location, human_goal, obstacle_goal)
        end if
    end if
end while
```

We have incorporated Proportional-Integral-Derivative (PID) control mechanisms into our system to enhance its robustness and stability. This implementation involves two distinct PID controllers, each strategically designed to minimize both linear and angular errors. By utilizing PID control, we aim to optimize the response of the system, effectively reducing discrepancies between desired and actual states. The P component tackles the current error, the I component addresses accumulated past errors, and the D component considers the rate of change of the error. The control law for a PID controller is listed below in the equation.

$$u(t) = K_p \cdot e(t) + K_i \cdot \int_0^t e(\tau)\, d\tau + K_d \cdot \frac{de(t)}{dt} \tag{1}$$

The flowcharts below explain the linear and angular PID control structure.

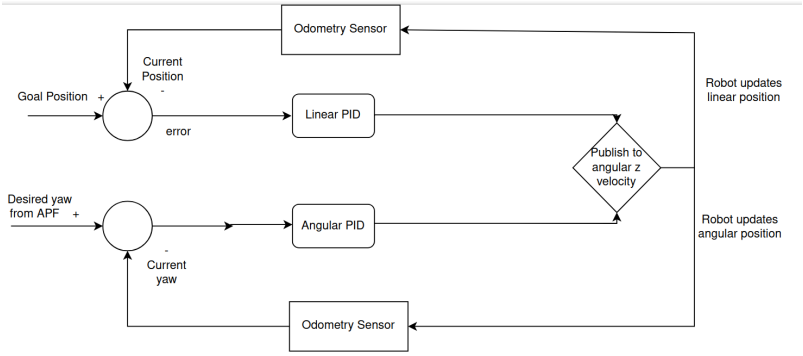


Figure 5: PID control structure

The linear output velocity

1. Proportional Error:
$$d = \sqrt{(linearerror_x)^2 + (linearerror_y)^2} \tag{2}$$

2. Integral Error: It is calculated by updating the distance at each interval. Where Dt is the linear error sum. δt is the difference between the current time and the last update time.
$$D_t = D_t + d * \delta t \tag{3}$$

3. Derivative error: It is the rate of change of error. Here d_p is the previous error.
$$D_d = \frac{d - d_p}{\delta t} \tag{4}$$

The final linear velocity output is obtained by summing up the proportional, integral, and derivative errors.

$$v(t) = K_p * d + K_i * D_t + K_p * D_d \tag{5}$$

Similarly, we can compute the orientation errors of the robot.

1. Proportional Error: For computing the angular velocity of the robot, we compute the angular error
$$e = yaw_{desired} - yaw_{current} \tag{6}$$

2. Integral Error: It is calculated by updating the angle at each interval. Where Et is the error sum. δt is the difference between the current time and the last update time.
$$E_t = E_t + e * \delta t \tag{7}$$

3. Derivative error: It is the rate of change of error. Here d_p is the previous error.
$$E_d = \frac{e - e_p}{\delta t} \tag{8}$$

The final angular velocity given to the robot is governed by the equation

$$w(t) = K_p * e + K_i * E_t + K_p * E_d \tag{9}$$

2.1.3 Sensor Fusion

We use two different sensor modalities for two different tasks. Hence to fuse the two modalities for our advantage we used Decision Level sensor fusion.

As can be inferred from Fig(8), the robot makes decisions as follows. When the robot is not in proximity of any obstacle, it tries to follow the human instructor, and as soon as it comes in proximity of the obstacle, it saves the last known location of the human instructor in the world coordinate frame and uses it as GOAL location for the obstacle avoidance algorithm. After the robot reaches the goal position again, it looks for the human instructor and continues to follow.

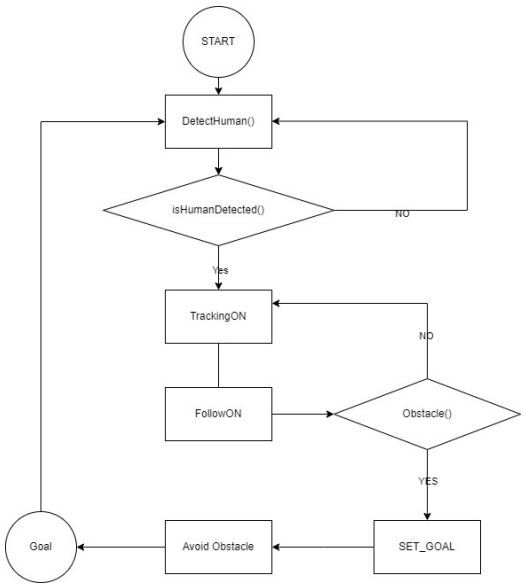


Figure 6: Fusion of both Camera and LiDAR

3 Results and Discussion

3.1 Human following robot experimental results

In order to verify the proposed method, we conducted two experiments by using a laboratory mobile robot. An RGB-D camera was mounted at the front of the robot for human-following and obstacle avoidance. Several interesting experimental results are presented to show that the robot can follow the user continuously while avoiding unexpected obstacles.

We have implemented the human following robot in the Gazebo environment using the ROS framework. Turtlebot3 with Intelrealsense R200 camera for depth sensing and LIDAR, for environmental awareness is used to perform our experiments. We tested our human following algorithm in 2 gazebo worlds. One world is the factory world to replicate the industry setting and the other is the cafe world.

(a) Cafe world (b) Factory world

Figure 7: Gazebo worlds used to test

The human in this experiment is an animated character that walks randomly in the environment and the robot follows the human. As the human walks in the corridor, the robot recognizes the specific user in the camera FOV with the pre-registered user data. The robot regarded the person with the highest similarity score and started to follow. The human detection results form two different worlds are shown in Figure 8.

(a) Cafe human detection (b) Factory human detection

Figure 8: Gazebo worlds used to test

While the robot is following the human and an obstacle is detected nearby the obstacle avoidance algorithm starts creating a safe trajectory for the robot. We computed the constants K_p,K_i,K_d by the hit and trial method. Figure 5 graph shows the change in final angular error for different K_p,K_i,K_d with time. The suitable values are $K_p = 0.06$, $K_i = 0.0$, $K_d = 0.01$

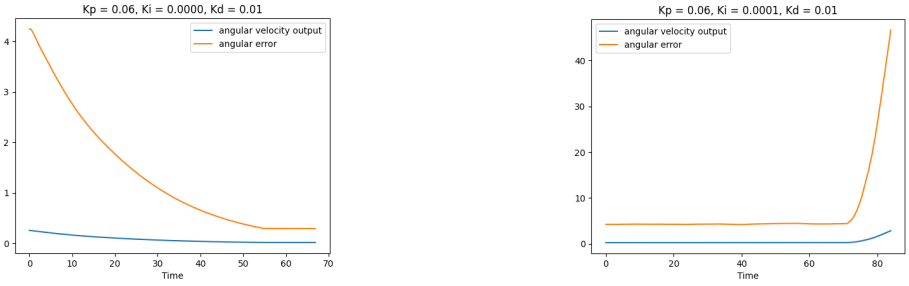


Figure 9: Angular error and velocity output VS time for different Kp,Ki,Kd

Similarly for linear error, the suitable values are $K_p = 0.06$, $K_i = 0.0$, $K_d = 0.01$

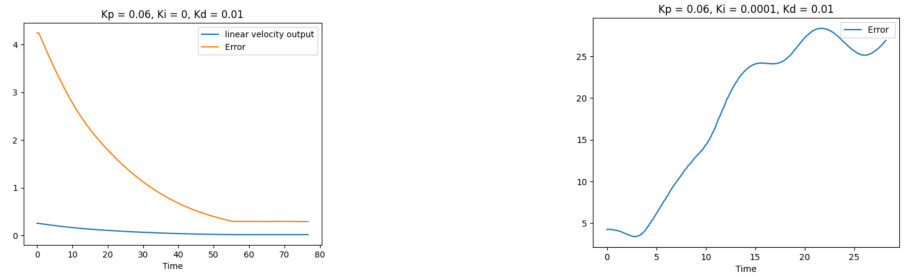


Figure 10: Linear error and velocity output VS time for different Kp,Ki,Kd

The tolerance for linear error is ±0.3m, and for angular error is ±0.2rad. When the robot enters the obstacle region without the human in its frame, it will autonomously navigate to the last known location of the human. This behavior ensures that the robot actively seeks the human even when not currently visible.

3.2 Limitations and Future Scope

While working on the project, we figured out the following limitations.

- The LiDAR sensor fails to detect objects that are too low for the robot. For Example, the base of a table. This causes a collision between the robot and the table.
- The tracking algorithm does not track a human when he moves too suddenly. In this case, the human has to move to the frame again so that the robot can detect him again, and the tracking can be continued.
- The robot has to look for the human again if he leaves the frame while the robot is avoiding the obstacle.

  Future developments for the human-following robot project may involve enhancing real-time processing capabilities, incorporating advanced sensor technologies for improved obstacle detection, and refining the algorithm to handle dynamic environments seamlessly. Additionally, exploring applications in diverse settings, such as crowded public spaces or collaborative industrial environments, presents promising avenues for further research and innovation.

4 Conclusion

In the realm of robotics and autonomous systems, the ability to interact seamlessly with human-centric environments is steadily gaining significance. This project aimed to develop robots capable of autonomously tracking humans by utilizing a combination of cameras and LiDAR sensors, promising a significant impact across sectors like healthcare and logistics.

Our algorithm integrates data from both LiDAR and camera sensors to achieve robust and dynamic human tracking capabilities, enabling the robot to navigate autonomously across diverse environments. This breakthrough contributes to advancements in human-robot interaction and holds promise in fields such as medical assistance, personal robotics, and warehouse automation.

Our ongoing work involves refining an autonomous obstacle avoidance algorithm using LIDAR data, while our immediate focus is on implementing a robust control algorithm. This control algorithm ensures the robot can follow humans safely, furthering the potential for human-robot collaboration in practical applications.

References

[1] M. Eisenbach, A. Vorndran, S. Sorge, H. Gross. User Recognition for Guiding and Following People with a Mobile Robot in a Clinical Environment. *Proc. of 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, **2015**, pp. 3600-3607.

[2] B. X. Chen, R. Sahdev, J. K. Tsotsos. Person Following Robot Using Selected Online AdaBoosting with Stereo Camera. *Proc. of 14th Conference on Computer and Robot Vision (CRV)*, **2017**, pp. 48-55.

[3] M. Gupta, S. Kumar, L. Behera, V. K. Subramanian. A Novel Vision-Based Tracking Algorithm for a Human-Following Mobile Robot. *IEEE Transactions on Systems Man and Cybernetics: Systems*, **2017**, **47**(7), pp. 1415-1427.

[4] Q. Ren, Q. Zhao, H. Qi, L. Li. Real-time target tracking system for person-following robot. *2016 35th Chinese Control Conference (CCC)*, Chengdu, China, **2016**, pp. 6160-6165, doi: 10.1109/ChiCC.2016.7554324.

[5] M. I. J. Salman Afghani. Follow Me Robot Using Infrared Beacons. *Acad. Res. Int.*, **2013**, **5**(1), pp. 59.

[6] J. Cai, T. Matsumaru. Human Detecting and Following Mobile Robot Using a Laser Range Sensor. *J. Robot. Mechatron.*, **2014**, **26**(6), pp. 718-734.

[7] C. -A. Yang, K. -T. Song. Control Design for Robotic Human-Following and Obstacle Avoidance Using an RGB-D Camera. *2019 19th International Conference on Control, Automation and Systems (ICCAS)*, **2019**, pp. 934-939, doi: 10.23919/ICCAS47443.2019.8971754.