Evaluation of the Performance of a Human Following Robot Under Varying Illumination

Rahul Krishnamoorthy
Department of Electrical and Computer Engineering
University of California, Davis
Davis, California, USA
rkrishnamoorthy@ucdavis.edu

Vishal I B
Department of Electrical and Computer Engineering
University of California, Davis
Davis, California, USA
vib@ucdavis.edu

***Abstract*—A meticulous and robust system for object detection and tracking is still a milestone in the field of computer vision. The various factors that hinder the accuracy of the model includes illumination, noise in the scene, occlusion effect and pose variations of which the source of illumination and its orientation with respect to the object plays a pivotal role. An illumination variation may result in the tracking algorithm to lose the object in the scene. This paper proposes a simulation of a robot which follows a human under varying illumination induced by varying heights and locations of the illuminating source with respect to the human. The camera on the robot is aided by a histogram of oriented gradients (HOG) approach with a Support Vector Machine (SVM) classifier for detection and tracking of the human in the scene.**

Keywords-Histogram of oriented gradients (HOG); Support vector machine (SVM); Human detection and tracking; Varying illumination;

I. INTRODUCTION

The moving object detection and tracking are important research areas for widespread application in diverse disciplines like the visual surveillance, human computer interaction, driving assistance system, autonomous navigation, traffic flow analysis and so on. In the last two decades, there have been several researchers who have devoted themselves to investigating different methods for solving the tracking and detection problem. These methods can be categorized according to the following issues: how to characterize the object and how to track them among the multiple frames. Human detection and tracking are tasks of computer vision systems for locating and following people in video imagery. Human detection is the task of locating all instances of human beings present in an image. Detecting humans in images is a challenging task owing to their variable appearance and the wide range of poses that they can adopt. Human tracking is the process of temporarily associating the human detections within a video sequence to generate persistent paths, or trajectories of the people. One of the main concepts in visual robotics is Illumination. The Illumination problem is defined as the degree of visibility of the object or change in appearance of the object with different lighting condition. The placement of the light sources can make a difference in the type of any object that is being presented. The variation of the position of light sources might create shadows of the robot and the target depending on the position of the light which will hinder the performance of the vision sensor. The variation in position of the light sources can be along any of the three axes with respect to the human and robot. This in turn affects the performance of the detection algorithm and the operation of the robot. The vision sensor or camera like the human eye requires light bouncing off objects to detect or recognize them. So, in the absence of light the sensor will not be able to detect objects. Outdoor surveillance systems or any outdoor vision system will be affected by the lighting conditions every day. The degree to which this affects the system can be calculated to make a robust system unfazed by the changing light conditions. The robot should be able to mitigate the losses in detection. Outdoor surveillance systems will also suffer from the change of weather conditions. Hence, they will reduce the performance of object detectors and trackers. In this paper we provide an analysis of the effect of different orientations of the illumination source on the detection performance of the robot.

II. LITERATURE OVERVIEW

A wide range of literature study presents several implementations of the tracking and following application. These implementations vary from one environment to another with increasing variables in the environment. A mobile robotic system for tracking and following moving people provides important capabilities for human robot interaction and assistance of humans in various settings. That is why tracking with mobile robots has been an active research area with many successful systems developed. In an outdoor, potentially cluttered, and dynamic environment where the sensors are subject to more noise, the tracked person can be occluded or lost for short periods of time. Typical applications include disaster-rescue missions or military mules carrying equipment and assisting soldiers. Tracking under the above requirements is harder and often, systems proved to work well normally are ineffective in such adverse conditions. [1] describes building two such systems, each having its own benefits and shortcomings. The first system detects and tracks motion from visual input only and uses a laser rangefinder to estimate the range to the centroid of motion in the camera image, while the second approach uses the laser as the primary sensor and camera as secondary. The first approach uses ego motion compensation in transformed perspective images and frame differencing to detect motion. The second system uses the laser to extract the 3D relative position of blobs that might have originated from a person and uses these as measurements to a probabilistic data association filter (PDAF).

During the past few years, several methods have been proposed in the field of object detection and tracking. Even though there are a plethora of algorithms, one of the robust and profound techniques is by using histograms of oriented gradients. Various studies have been made to study the performance of model by varying the intensity and orientation of light source during object detection and tracking. One of the approaches is a discrete wavelet transformation for detecting and tracking moving objects under varying illuminations with a stationary camera [2]. Discrete wavelet transformation provides a method of illumination invariant feature extraction which makes use of gaussian smoothing and thresholding. This method can detect and track non-rigid moving objects, even when light intensities change abruptly. Discrete wavelet transforms (DWT) was used to provide multi-resolution images and to decompose the original image into the sub band images including low-and high-frequencies. Discrete wavelet transforms were used to compute accurate and efficient estimate of the object structure in each video frame, mainly for the edges of the object. Initially the video was converted into the frames and on each frame, first level wavelet decomposition was performed using Daubechies second order filter. Non-rigid object tracking using mean shift algorithm is implemented on the video datasets. Mean shift is a hill climbing algorithm which involves shifting the kernel iteratively to a higher density region until convergence. Every shift is defined by a mean shift vector. The mean shift vector always points toward the direction of the maximum increase in the density. It is a basic algorithm that may be used in data clustering, analysis and computer vision applications. In this algorithm, tracking of non-rigid objects characterized by the color and/or texture is used as the object of interest. After the frames were processed by following all the above steps, the performance of the model was evaluated. It was concluded that the wavelet transform method provided one of the best results for object detection and tracking with abrupt changes in light intensities. They provided a performance review of the algorithm with varying intensities and it might be interesting to address the performance of the algorithm in various orientations of light with respect to the object.

Another robust motion detection method for illumination variations uses a histogram of oriented gradients [3]. The detection process that was used was divided into two phases: coarse detection and refinement. In the coarse detection phase, a texture-based background model is built which implements a group of adaptive histograms of oriented gradients. On comparing the histogram of oriented gradients of each pixel between the current frame and background model, a foreground can be segmented based on texture feature which was not susceptible to illumination variations. The result based on texture is optimized by combining the pixel-wise detection result produced by Gaussian Mixture Model (GMM) algorithm, which incorporates efficient morphological operations. In the next phase, which is the refinement phase, the above detection result was refined based on the distinction in color feature to eliminate errors like shadows, noises, etc. The experimental results showed a good detection performance against illumination variances. But again, the illumination variations are only based on the intensity of the received light and not on the orientation of the light source with respect to the object. By implementing this test, the model's performance can vastly differ.

The movement of the visual sensor system can be very useful in case of tracking the non-linear motion of the target. Visual tracking system of a humanoid robot is discussed in [4]. When observing and tracking a moving target, the head and neck mechanism of humanoid robot adjusts the position and orientation of the visual perception module to make it always aim at the target. The motion mechanism of visual tracking is designed to imitate the human head and neck, which is a 3DOF manipulator consisting of lower pitching joint, yaw joint and upper pitch joint. The modular joint design method is adopted to reduce the design complexity. When the human tracks a target, the method is to search the target in a wide area and locate it roughly firstly, and then take a closer look. The visual tracking process of humanoid robot is like the human. The search and location of target is carried out from the macro to the micro area by using two types of cameras that are TOF camera and CCD camera respectively.

A system consisting of several modules that can communicate with each other under the Robot Operating System (ROS) framework is presented in [5]. The main modules of the system are an object tracker and an Image Based Visual Servoing (IBVS) controller. This tracker can robustly track objects on the drone's video stream. A great advantage of object trackers with learning capability is that they do not require any previous knowledge of the tracked object. The IBVS controller closes four feedback loops based on image features, which are the bounding box location and size. The references to the controller are the desired location of the centroid of the bounding box in the image, and the size of the bounding box. The resulting behavior of the system is that the drone will turn to look at the target and approximately control its relative position with regards to the target. The image based visual-servoing method that uses only a forward-looking camera for tracking and following objects from a multi-rotor UAV, without a dependence on any GPS system. This method tracks a user specified object continuously while maintaining a fixed distance from the object and simultaneously keeping it in the center of the image plane. The experiments show that the system can track a great variety of objects present in suburban areas, among others: people, windows, AC machines, cars and plants.

The solution proposed in [6] addresses the problems of people detection and tracking by mobile robots in indoor environments. A system that can detect and recognize people is an essential part of any mobile robot that is designed to operate in populated environments. The main challenges for people tracking systems come from the fact that people have articulated bodies and their appearance can change drastically depending on pose, view, clothes, self-occlusions of different body parts, etc. Moreover, their behavior can be very unpredictable. Therefore, creating an effective model of human appearance and behavior can be a very complex task. By contrast, tracking of rigid and predictable objects such as cars is much easier, which has resulted in many successful existing applications. The system presented in is non-invasive, since the only sensors used are thermal and color cameras. It is robust, due to the use of a probabilistic tracking algorithm. Information provided by sensors can be imprecise or even misleading due to the sensor noise, clutter and dynamic occlusions caused by other objects or persons. Therefore, to reliably estimate the location and movement of persons it is necessary to apply a tracking procedure. Tracking also enables combination of information from different sensors, giving more accurate and complete results. The most popular approach to the tracking problem is based on the state space representation. Following this method, a person's kinematics can be described by a state vector and create a dynamical model of the person's movement. Tracking in this case is equivalent to the state estimation problem for a dynamic system given, the sensor observations. This work makes use of Bayesian inference, a widely accepted framework within the tracking community that models uncertainty in the system by means of probabilities.

The automotive industry is increasingly interested in adding more intelligence to cars and trucks with the goal of developing fully autonomous automobile traffic. To this end one of the most important research areas to address is that of automated perception systems that will allow the vehicle to comprehend its immediate environment and make decisions. The change in appearance of the target with viewpoints poses a challenging task in detection. [7] discusses the detection of bicycles that share the road with intelligent vehicles. Bicycle detection is important because bicycles share the road with vehicles and can move at comparable speeds in urban environments. From a computer vision standpoint, bicycle detection is challenging as bicycle's appearance can change dramatically between viewpoints and a person riding on the bicycle is a non-rigid object. A vision-based framework is presented to detect and track bicycles that take these issues into account. A mixture model of multiple viewpoints is defined and trained via a Support Vector Machine (SVM) to detect bicycles under a variety of circumstances. Each component of the model uses a part-based representation and known geometric context is used to improve overall detection efficiency. An extended Kalman filter (EKF) was used to estimate the position and velocity of the bicycle in vehicle coordinates.

A mobile robot system where the robot tracks a moving target was presented in [8]. The system minimizes the probability of losing the target. If the next position of a moving target has the Gaussian distribution, the proposed system guarantees the tracking success probability and the moving distance of the mobile robot is minimized based on the chosen bound on the tracking success probability.

The performance evaluation should be quantitative. It should report how many objects were detected correctly and how many false positives (false alarms) were produced. It should scale up to larger test areas or multiple 3D scenes without losing its tracking capability. There are three main types of performance metrics in the system proposed in [9]: detection-based metrics; tracking-based metrics; and perimeter intrusion detection metrics. The detection-based metrics are used to evaluate the performance of a System Under Test (SUT) on individual frames from video sensor data. The tracking-based metrics use the identity and the complete trajectory of each object separately over the test sequence and compare the Ground-Truth (GT) tracks with the SUT tracks based on best correspondence. Then, based on the best matches, various error rates and performance metrics are computed. Finally, the perimeter intrusion detection measure is based on detecting any object when it enters a specified area. The methods used for determining object correspondences between the SUT's data and the GT data significantly affect the values of the performance measures. The main criteria are: 1) Object matching based on the object area intersection criterion is measured by calculating the overlapping area of the SUT-reported bounding box with the GT bounding box at each frame, with a threshold selected for a successful match. 2) Object matching using object centroids is based on measuring the Euclidean distance between the object's centroid as reported by the SUT and the GT data at each frame, with a threshold selected for a successful match. Normalization based on the size of the bounding box is used often. Two measures are used to express the performance of the tracker. The first is the tracking precision, which expresses how well the tracker estimates the exact positions of objects or people. The second is the tracking accuracy, which measures how well the system keeps track of people or objects and how many mistakes are made in terms of misses, false positives, mismatches, failures to recover tracks, etc.

III. Problem Definition

How does the variation of the spatial arrangement of the illumination source affect the performance of the human following robot?

The simulation environment under consideration is an outdoor space and the illumination source is the sun. By performance we refer to the accuracy with which the robot detects and tracks the humans under different arrangements of the light source. With varying light source orientation, the intensity and the angle of incidence of the light varies and this in turn affects the performance of the robot. The human does not move in a straight line and is going to be moving in a fixed trajectory. The robot tracks and follows the human along the trajectory maintaining a certain distance. By placing the light source at different angles with respect to the vertical plane of the environment, we observe the changes or hindrance to the performance of the robot. We train a support vector machine (SVM) to classify between humans and the background and to create a bounding box around the human if the classifier has detected one. We obtain the confidence scores from the SVM classifier for each frame and compare these against a predetermined threshold value to decide whether the object is being detected or not. The number of times ('miss-count') the confidence score value falls below the threshold can be used as a metric to determine the failure in detection due to varying illumination. Another metric we use to determine the performance of the robot is by observing the distance travelled by the robot during one iteration of the simulation.

IV. Simulation

Simulation for our research question requires the robot and the environment to be as close to reality as possible. The robot, the human, and the lighting conditions must be real, that is, obey the rules of physics in order to consider it as an analysis. Gazebo along with ROS and MATLAB serves this purpose and these are the simulation software that are going to be used here. Gazebo is a 3D dynamic simulator with the ability to accurately and efficiently simulate population of robots in complex indoor and outdoor environments. Gazebo contains a wide variety of models, fitted with a physics engine, required for the simulation. ROS is an open-source, meta-operating system for robots. It provides the services expected from an operating system, including hardware abstraction, low-level device control, implementation of commonly used functionality, message-passing between processes, and package management. MATLAB provides the required computer vision, navigation and robotics toolboxes which can be used to manipulate the robot. Combining these three platforms gives a very powerful robot simulator.

A. The Robot

The process of object detection and tracking for humans is a computationally intensive task. This requirement paved the way to choosing Turtlebot 2.0 as the robot of our choice. The Turtlebot has an Intel Core i3-4010U which will help us do complex calculations efficiently. The robot's intel HD graphics driver makes it one of the best choices for computer vision tasks. It also has 4 gigabytes of memory with a 500 GB of internal hard drive which helps us store lots of images for the processing.The robot has dimensions of 14.0 x 14.0 x 16.5 inches which is required to hold all the memory and computation power. The robot weighs 6.3 kilograms which further adds on to the sturdiness of the robot. The Turtlebot 2 comes with a Kobuki base on which it is mounted. The Kobuki base has two drive wheels to make translational and rotational motions along with two passive wheels to reduce the rocking of the robot. All wheels have a diameter of 7.6cm.

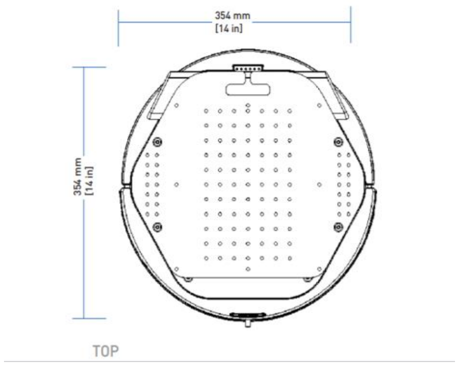

Fig 2. The above picture shows the top view of the robot along with its dimensions.

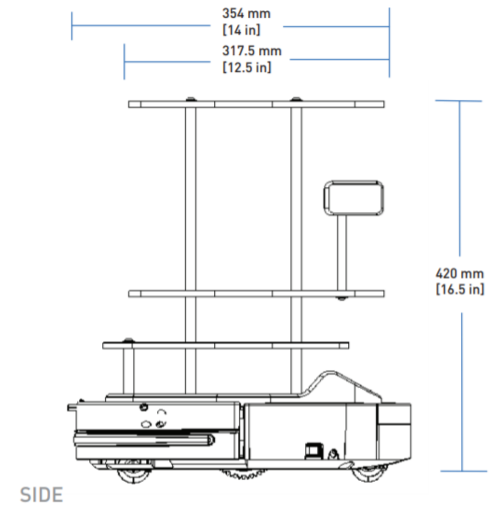

Fig 3. The above picture shows the side view of the robot along with its dimensions.

We assume there is no slipping of wheels on the floor to reduce the complexity of the scene. The robot has a maximum translational speed of 65 cm/sec. The speed with which the robot is moving at a given time can be measured by using an odometer. The odometer can measure up to a precision of 11.7 ticks per millimeter of the encoder in the odometer. Gazebo allows us to specify the speed indirectly by giving the distance to be travelled and the time taken to cover that distance for a fixed trajectory. With the distance and the time, we can determine the average speed and the human is made to move with this calculated speed constantly throughout the trajectory.

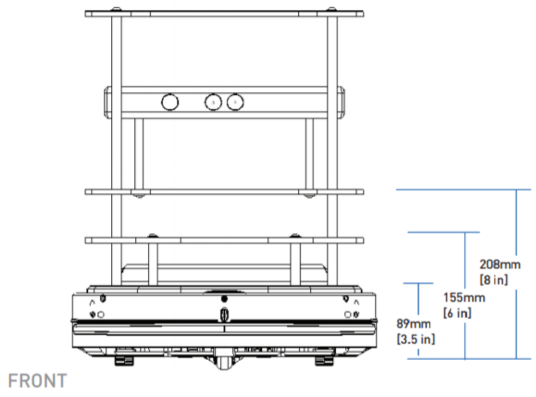

Fig 4. The above picture shows the front view of the robot along with its dimensions.

Thus, the robot and the human have a constant speed of 50 cm/sec throughout the simulation. The motors used for movement are brushed DC motors and they have a H-bridge voltage source in order to switch the polarity. The speed of the robot can be controlled by using pulse width modulation (PWM). Since the speed is analogous to the power the robot receives, pulse width modulation can be an effective method which cuts the power into discrete parts and thus reducing the average power it receives. But since we are going to be driving the robot at maximum speed, this feature is not necessary. It also has a 3-axis digital gyroscope to measure the orientation of the robot. It has a maximum range of ±250 degrees/sec and the yaw axis is factory calibrated within the range of ±20 deg/s to ±100 deg/s and hence the orientation of the robot with respect to the human can be calculated. The motors run in the same direction with the same speed while moving straight and when the robot wants to turn in any direction, it makes use of differential drive. During differential drive, if the robot wants to turn left, the left motor and hence, the wheel start moving in the reverse direction while the right side continues to move in the same direction and vice versa for the right side. Above the Kobuki base are the Turtlebot module plates where on the first floor, Turtlebot netbook along with a router is placed.

The Kobuki base implements the task that is given by the netbook. The router is connected to the netbook so that the robot can transmit and receive information to and from the workstation using local IPs. On the next floor is the Microsoft Kinect camera which serves as the vision sensor in our application. The Kinect camera is attached to the netbook using it's mounting hardware.

B. Environment

The environment in the simulation includes everything the robot interacts with. The purpose of the simulation is to evaluate the performance of a human following robot with varying spatial arrangement of the illuminating source. The environment in which this analysis is done is created with a few assumptions so that the performance can be studied clearly. There will not be any kind of obstacles in the entire region of simulation except the human. It is also important to consider that the entire simulation does not have more than one human. The human always moves through a fixed trajectory during the simulation. We have decided that the trajectory should be a quadrilateral which does not have any two sides parallel so that symmetry in the system is eliminated. It is essential that the human moves through the same path every time so that the human following robot can be clearly analyzed under varying lighting conditions. The human in the environment takes strides of the same length during the entire time of the simulation and the pace which he moves remains constant throughout. While the human along with the robot moves along a fixed trajectory, the light source stays fixed during one iteration of the simulation.

The spatial orientation of the illuminating source is labelled on the cardinal system. The light source is placed at 13 different locations spread out in the x-y plane and at three different heights, ground (zero meters), height of the human (1.9 meters), and twice the height of the human (3.8 meters). The performance of the model is recorded for every location.

C. Architecture

Historically, there are three main paradigms for a robot architecture - Reactive, Deliberative, Hybrid. A reactive architecture does not have a planning phase and directly links from the sense phase of the model directly to the act phase. This architecture is not suitable for our application as the robot must understand the environment, detect the presence of a human, move towards the human and constantly keeps tracking the human at equal intervals. Since, the sensing and the planning part alone is not enough for accomplishing this task, we move towards a deliberative architecture. The robot implements detection, tracking and following functions. We implement each of these functions in different ROS modules(nodes). These nodes communicate with each other through ROS messages. Any deliberative system includes three phases: Sensing, Planning, Action.

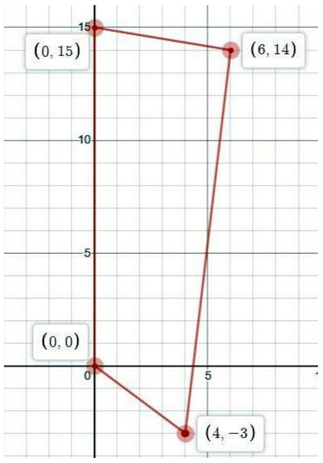

Fig 5. The quadrilateral shows the fixed trajectory of the human.

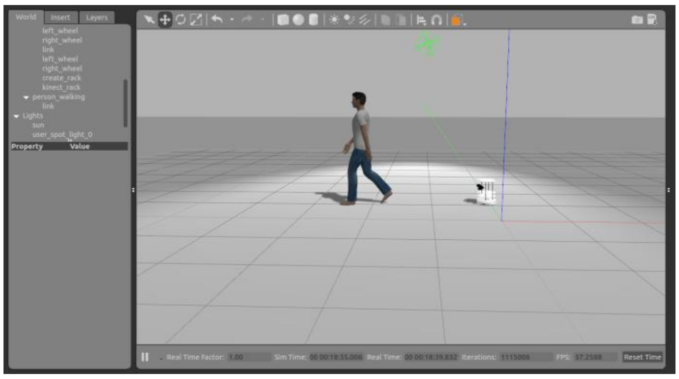

Fig 6. Picture shows how the human, the robot and the illuminating source looks like in the Gazebo simulation.

1. SENSING

The simulation starts with a human standing 3 meters away from the robot. Since this distance is below its optic range, the robot will not be able to detect the human. As the human slowly starts walking away from the robot at a constant velocity of 0.50 meters/sec in a fixed trajectory as shown in the figure, the human falls into the optic range of the robot's vision sensor and hence the sensor will be able to detect the human.

The Kinect camera has an RGB camera as well as an IR depth sensor. Both have a horizontal field of view of 57 degrees and a vertical field of view of 43 degrees. The cameras have an effective range of 0.5 meters to 6 meters below and above which the object won't be focused properly. The human is simulated to have a height of 1.66 meters. Since the vertical field of view is 43 degrees, the human must be at least 3.42 meters away from the robot for it to detect properly. Moreover, the upper limit of the effective range for focusing is 6 meters, hence the human always must be at 3.42 meters to 6 meters from the robot. If the robot loses track of the person and if the person walks more than 6 meters away from the robot, the robot will lose track of the person completely and will not be able to recover. In order to increase the range further, we propose to tilt the camera by 8.5 degrees in the vertical direction. This reduces the minimum distance at which the human will be detected from 3.42 meters to 2.34 meters. The Kinect RGB camera has a resolution of 640x480 and has a 24-bit color range. The camera can capture up to 30 frames per second, but we do not need that much data for our problem. It is enough if we take 4 frames per second and skip all the other frames. This is because we have specified that the human moves at a constant pace of 50 cm/sec and hence between each frame, the person moves only 1.25 meters away from the camera which is well between the effective optical range of the camera.

2. Planning

As the human moves from one place to another the robot takes frames of images and processes them to provide accurate detection. For the robot to be able to detect the human, it requires a feature extractor and a classifier that must be trained. When an image is input to the feature extractor, it extracts informative and non-redundant features from the image which can be used for further processing. The histogram of gradients (HOG) features extractor can be trained on robust structural cues like shape instead of color as we will be testing our robot in conditions where there is going to be minimal illumination. Instead of operating on the image pixel by pixel, we combine the pixels to form cells. For each cell, we compute all the directions of gradients and group them into several orientation bins and then sum up the gradients in each sample. What this means is that the gradients which are higher are taken to be stronger and they contribute more towards the weight to their respective bins. Another advantage of this mechanism is that it reduces the effect of small random orientations due to noise in the image. This way of storing the features helps in keeping the representation of the object distinct while allowing small variations in the shape. We should give the HOG feature extractor the number of orientations, pixels per cell, and ells per block as parameters to compute the HOG features of a single channel of an image. The number of orientation bins that are required for the gradients of pixels in each cell to be split up in the histogram. The pixels per cell number specifies the number of pixels in each row and column per cell for each gradient in the histogram. The local area over which the histogram takes a normalized count in each cell is given by the number, cells per block.

Following the training of the feature extractor, the classifier must be trained. An image classifier is a model that has been trained on multiple image classes and given an image it provides us with the class to which the image belongs. For this purpose, we used the ‘UprightPeople’ model from MATLAB that has been trained on images of size 128x64. This model contains only two different classes, humans and the background. The images used to train the model includes background pixels around the person. Therefore, the actual size of a detected person is smaller than the training image size. This is necessary for the classifier to understand how the human looks like when the robot is following the human. The penalty attribute of the SVM classifier can be tuned such that we obtain the maximum accuracy. Before we deploy the model in the robot simulation, we must optimize our detection phase. The HOG feature extracting phase is vastly time-consuming. So instead of computing the HOG features on the patches of images that tend to overlap, we extract HOG features of the whole frame at the beginning, then we pull out the features for each frame’s subregion as we need them. The SVM classifier now outputs the class scores for each class (human and background) and the bounding box coordinates if it detects a human.

Now there will be multiple bounding boxes obtained for the same object because of the optimizing algorithm used for HOG and these duplicates must be removed. For this purpose, we use an algorithm known as non-max suppression. This algorithm computes the intersection over union for this purpose. It is defined as the ratio of intersection of the area of the bounding boxes to the union of the area of the bounding boxes. If the Intersection over Union (IOU) is a great value, then it means that the bounding boxes are representing the same object and hence the redundant bounding boxes can be removed. For this purpose, we choose the bounding box with the largest score and we compute the IOU with other bounding boxes and remove all the bounding boxes which have an IOU greater than 0.5. Now the bounding box coordinates represent the object (human) uniquely.

These bounding box coordinates are present in the image frame of the camera. That is, these are basically pixel coordinates. These coordinates can be used to determine the pose of the human at each step. The calculated pose refers to the position and orientation of the human in the image frame. The estimated pose can be converted from the camera frame to the map frame using various transformations. But we decided to use the coordinates in the image frame. Our algorithm works with the points in the image frame i.e., the pixel coordinates. The objective is to align the robot with the human in the vertical axis so that the robot is always facing the human. Since the robot moves only in the 2D plane, alignment in the horizontal axis is not necessary. The center of the image (IC), which is the center of the camera, should be aligned with the center of the human. We assume the center of the camera to approximately coincide with the center of the robot. The center coordinates of the human (HC) can be obtained from the coordinates of the best fitting bounding box. We then try to minimize the horizontal distance between the points HC and IC so that they are aligned along the vertical axis. The robot should be given an appropriate angular velocity command such that the point IC aligns with HC at every instant.

3. Action

The Turtlebot is given a linear velocity of 0.6 meters/sec, to compensate for the time lag caused by the detection algorithm, once it starts to detect the human. We use an approximate linear transform to map the angular displacements of the robot with pixel distances in the image. Based on this, the angular velocity commands are given to the Turtlebot at every instant to align itself in the direction of the human. Ideally, the robot follows the human at a certain distance in such a way that the human is within the field of view of the camera. In an event where the robot loses track because of a failure in detection either due to the human moving out of the frame or the robot moving very close to the human, the robot stops moving, rotates at an angular velocity of 0.2 radians/sec in the clockwise direction while sensing through the rotation to detect the human. Once it detects the human, the angular velocity is set to zero and the robot is given a linear velocity. One iteration of our experiment is completed when the human reaches the starting point after traversing through the trajectory. According to our experiment, we vary the lighting conditions for every iteration and observe the false negatives in the detections. We will be implementing different nodes for detection, tracking and following in ROS. Each node will publish and subscribe to various ROS topics.

D. *Experiment*

To investigate the performance of the Turtlebot under different lighting conditions, we defined a trajectory for the movement of the human. In order to compare the performances under different lighting conditions, the trajectory taken by them human should remain the same for all the scenarios. We conducted the analysis for five different scenarios.

1. Scenario 1

In the first scenario, the environment contained the human and the Turtlebot under the illumination of the sun along with the background light and the ambient light. The sun is basically a spherical bright light source. It was placed at the origin at a height of 10 meters from the ground. The human was moved along the pre-defined trajectory, and the performance of the robot was evaluated using certain performance metrics.

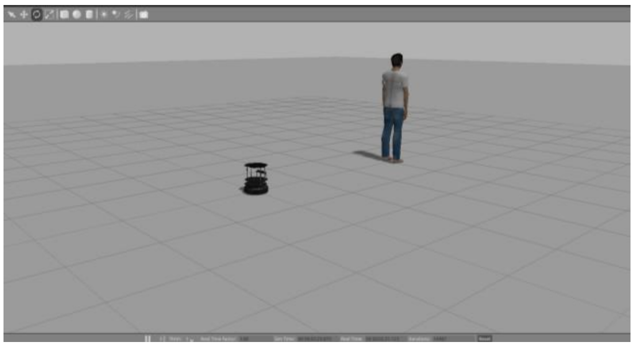

Fig 7. Well-lit environment with the sun and background light.

2. Scenario 2

Now, we wanted to test for the worst-case scenario. In the second scenario, all the light sources, including the background and the ambient lights are removed from the environment. The environment contains the human and the Turtlebot and it is completely dark.

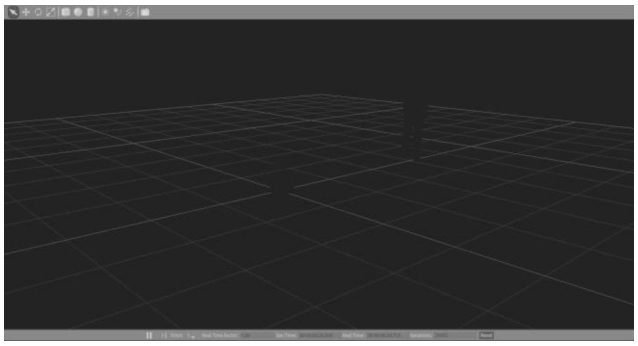

Fig 8. Environment with all the illumination sources removed.

3. Scenario 3

In order to find the optimal light setting, in the third scenario, all the light sources were removed from the scene except a single light source. This scenario contains only one point light source without the background light and the ambient light. The light was placed at a different location for every iteration of the experiment. 13 different locations were chosen on the trajectory, and at each location the light was placed at three different heights (0, 1.9 meters, and 3.8 meters) to study the effect of height of the illumination source.

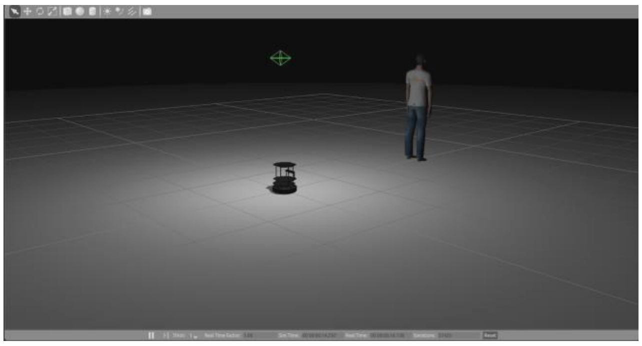

Fig 9. Environment with all the illumination sources removed except for a single point light.

4. Scenario 4

In the fourth scenario, all the light sources were removed except the background light. The experiment was performed with the human moving around along the trajectory and the robot following the person with only the background illumination.

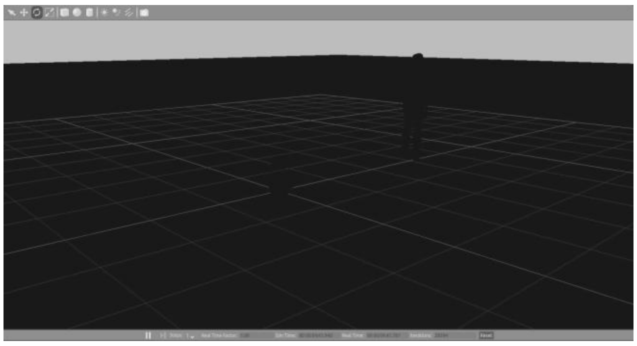

Fig 10. Environment with all the illumination sources removed except for the background light.

5. Scenario 5

In the fifth scenario, we wanted to test scenario 4 with an additional point light source. So, all the light sources were removed from the scene except for the background light and a single point light source. Like the third scenario, the point light is placed at 13 different locations and varying heights to analyze the performance of the Turtlebot.

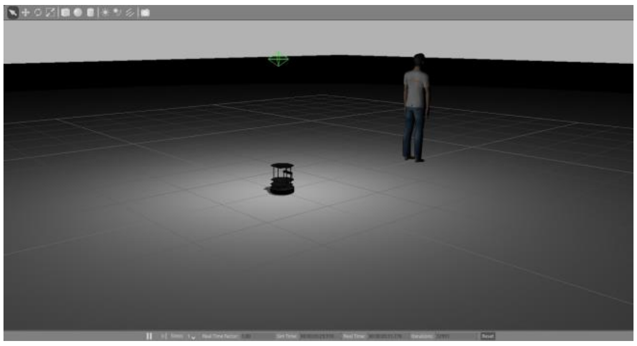

Fig 11. Environment with a single point light source and background light.

V. Simulation Results

The objective of this paper is to evaluate the performance of the human following robot under varying illumination. Two different metrics were used to evaluate this performance. The metrics are ‘Miss-count’ and ‘Distance Covered’ for each iteration of the simulation. Miss-count represents the number of times the robot fails to detect the presence of the human which is the number of times the confidence score falls below a threshold. The threshold was set to be 70% which means that whenever the surety of the robot finding the human is less than 70%, ‘Miss-count’ value is increased by 1. The robot was programmed in such a way that whenever the confidence score falls below the threshold, the program does not return the bounding boxes and creates a run-time error. To overcome this, a pair of ‘try-catch’ statements were used. These statements work together such that whenever there is a run-time error in the ‘try’ block, it breaks the code and jumps to the ‘catch’ block. This happens for each iteration of the simulation. We calculated the metric, ‘Miss-count’ on two different machines – Acer predator Helios 300 with an Nvidia GeForce GTX 1660Ti GPU, and HP Pavilion 15-au112TX with an Nvidia GeForce 940MX GPU. Surprisingly, the results weren’t consistent when we tried to run the simulations on these machines. There was a huge mismatch in miss-count values with a difference in the range of 100s. Having different processors, the machines had different execution times, and this caused the mismatch in miss-count values. The Acer predator having a faster execution time, had a miss-count of 344 and the HP-Pavilion had a miss-count of 125 for the same exact simulation. In order to avoid this mismatch and for our results to be consistent, we tested out all our simulations in just one machine, the HP-Pavilion laptop.

Moving on to the second metric, distance covered by the bot during each iteration of the simulation was calculated and used as a metric to compare how far the bot has travelled. The time intervals the algorithm spends during the try phase was computed and stored. All these time intervals were summed to get the total duration for which the bot moves during one iteration of the simulation. Once the total time spent is obtained, multiplying this total time with the velocity of the bot gives the distance covered by the bot for that iteration.

One iteration of the experiment is completed when the human reaches the starting point after traversing through the trajectory. According to the experiment, the lighting conditions are varied by changing the locations of these light sources along three different axes and the variation of the metrics in each case is observed.

1. Scenario 1

The first scenario is to observe the robot’s performance using the two metrics in the presence of sun illumination, ambient light along with background light. The robot is expected to track and follow the human through the entire path as this offers the best-case scenario for the robot in terms of illumination in the scene. As expected, the robot did follow the human for the entire path. It received a miss-count of 80 and travelled 36.76 meters. This gives the scores which is assumed for now to be the best-case scenario. On comparing these scores with the scores received from the other scenarios a conclusion can be made whether the robot is tracking the human well or not.

2. Scenario 2

The second scenario involves observing how the robot tracks and follows the human in the absence of all kinds of lights in the scene. Since this involves removing all sources of illumination such as sun illumination, ambient light and, background light, the robot failed to detect the presence of the human in the scene and hence failed to move. The robot received one of the highest scores for the parameter, miss count and received the least possible score for the metric, distance covered. The robot in this scenario received a miss-count of 192 and failed to move and hence, received a distance metric of 0. This scenario is the exact opposite of the previous one and it recorded the worst possible values for the metrics and if the metrics are comparable to these scores, we know that the robot is doing bad for that scenario.

Table 1 offers a comparison between how the metrics would look like for the best case (when the robot follows the human for the entire track) and the worst case (when the robot failed to move). This is useful to comment about the performance of the human following robot in any scenario by comparing with these values.

Name of the Scenario	Miss-Count	Distance covered
Scenario 1	80	36.76
Scenario 2	192	0

Table 1: Comparison of the results from scenario 1 and 2.

3. Scenario 3

In this scenario a single point source is kept in different places in the scene for different heights and the miss-count and distance covered metrics are calculated. Fig. 12 and Fig. 13 show how the miss-count and distance covered metrics change for the point light sources kept at different parts of the scene on the ground.

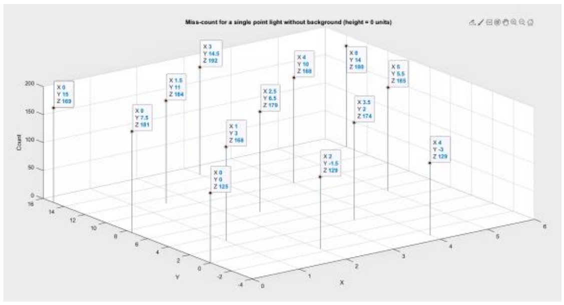

Fig 12. Miss-count for single point light without background (height = 0)

In the Fig. 12 and Fig. 13, the X-Y coordinates represent the location coordinates in the simulation environment and Z coordinates in Fig.12 and Fig.13 indicate the miss-count and the distance covered respectively. The stem graph is plotted on the X-Y coordinates indicating the position of the light source for that iteration of the simulation. Each stem indicates the position of the light source in the scene for that iteration in the absence of any other light source.

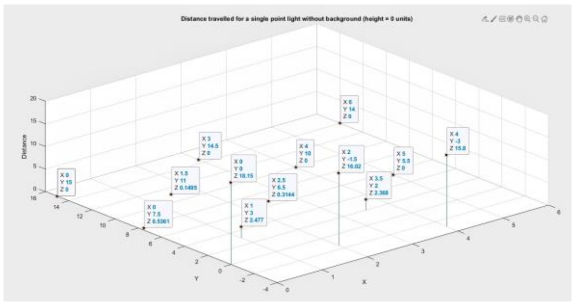

Fig 13. Distance travelled for a single point light without background (height = 0)

We have done the same experiment for the point light source at the same points along the scene but for different heights such as at the height of the human and for two times the same height. So, for the same X and Y coordinates, the point light was placed at three different heights (Z=0, 1.9, 3.8 meters). In order to compare these results efficiently, the miss-counts for all the scenarios were plotted in a single graph and likewise for the distance covered as well.

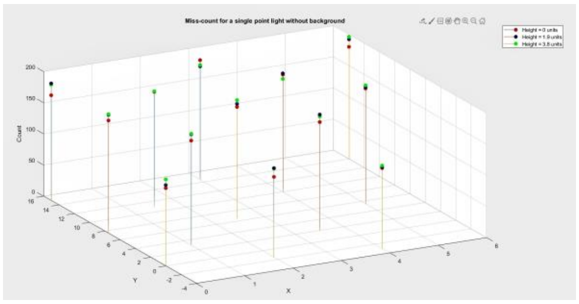

Fig 14. Miss-count for single point light without background

Fig. 14 and fig. 15 represents the consolidated graph for the miss-count and distance covered for different points with three different heights. Each stem with an X-Y coordinate represents the location of the point light source for that iteration of simulation. For a location on the X-Y plane there are stems which are coloured differently. There are three colours of stems in each location on the X-Y plane. The red colour stem indicates the miss-count and distance covered respectively for an iteration of the simulation when the point source is on the ground with the corresponding X-Y coordinate. Similarly, blue and green stems represent the metrics for Z = 1.9 and 3.8 meters respectively.

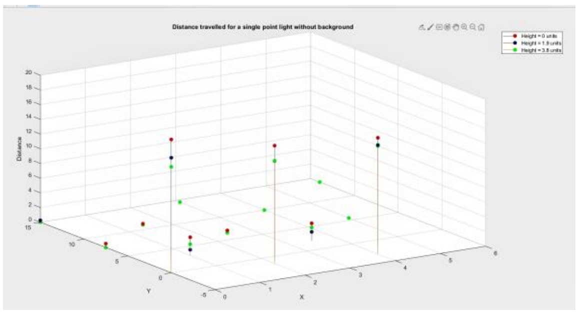

Fig 15. Distance travelled for a single point light without background.

From Fig. 15 it can be seen that whenever the X-value of the point source is lesser than or equal to 0, the robot tends to cover a longer distance than for other values. This is evident from Fig. 14 as well, where the miss-counts for all three heights are relatively lower for these cases. This is because whenever the X coordinate of the light source is lesser than or equal to 0, it is also lesser than or equal to the X coordinate of the human which is 0 when he starts his trajectory. This means that in all these cases, the point light illuminates the back of the human. When this happens, the human is visible well in the camera frame for the robot to identify the presence of the human and to follow him.

It can be seen from the individual stems from Fig 14. that the red stems tend to have a lower miss-count than blue and green stems. The red stem indicates that the point light is on the ground and the blue and green stems represent the miss count value when the point light is at a height of 1.9 and 3.8 meters respectively from the ground. This is also evident from Fig. 15 where the distance covered is relatively higher for red stems than the corresponding blue and green stems. This shows that whenever the point light is on the ground, the robot tends to follow the human better. The reason for this behaviour is explained in the discussion section.

4. Scenario 4

In this scenario, the simulation environment doesn't have any illuminating source except background light. We test this simulation to evaluate miss-count and distance covered. Surprisingly, the robot followed the human through the entire path with this minimal lighting. Table 2 represents the comparison of miss-count and distance covered for Scenario 1 and scenario 4.

Name of the Scenario	Miss-Count	Distance covered
Scenario 1	80	36.76
Scenario 4	70	37.10

Table 2: Comparison of the results from scenario 1 and 4.

Table 2 shows that the robot in scenario 4 did not just complete the entire track but outperformed the robot in scenario 1 by a huge margin. The miss-count is significantly lower for almost the same distance covered setting the benchmark as the best performing scenario.

5. Scenario 5

In this scenario, the performance of the robot in the presence of background light along with a point light placed at different positions and heights in the environment is observed. Fig. 16 and Fig. 17 show how the miss-count and distance covered metrics change when the point light sources are kept at different parts of the scene on the ground.

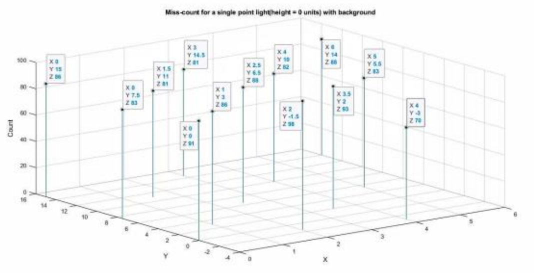

Fig.16: Miss-count for a single point light with background (height =0).

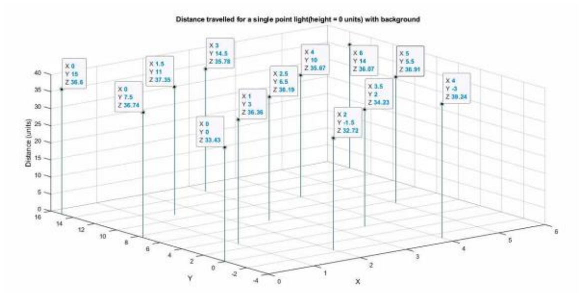

Fig.17: Distance travelled for a single point light with background (height =0).

In the Fig. 16 and Fig. 17, the X-Y coordinates represent the location coordinates in the simulation environment and Z coordinates in Fig.16 and Fig.17 indicate the miss-count and the distance covered respectively. The stem graph is plotted on the X-Y coordinates indicating the position of the light source for that iteration of the simulation. Each stem indicates the position of the light source in the scene for that iteration in the absence of any other light source.

The same experiment was performed for the point light at the same points along the scene but for different heights such as at the height of the human and for two times the same height. So, for the same X and Y coordinates, the point light was placed at three different heights (Z=0, 1.9, 3.8 meters). In order to compare these results efficiently, the miss-counts for all the scenarios were plotted in a single graph and likewise for the distance covered as well.

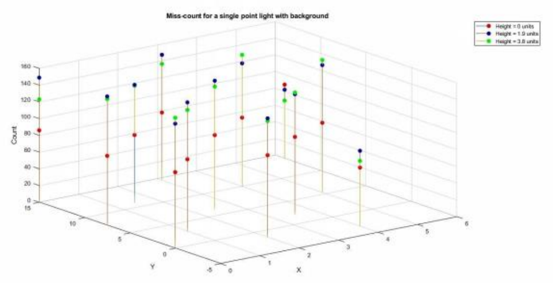

Fig. 18: Miss-count for a single point light with background.

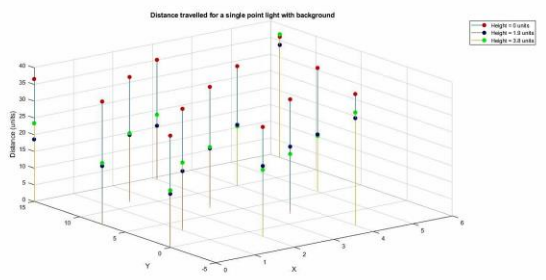

Fig.19: Distance travelled for a single point light with background.

Fig. 18 and fig. 19 represents the consolidated graph for the miss-count and distance covered for different points with three different heights. Each stem with an X-Y coordinate represents the location of the point light source for that iteration of simulation. For a location on the X-Y plane there are stems which are coloured differently. There are three colours of stems in each location on the X-Y plane. The red colour stem indicates the miss-count and distance covered respectively for an iteration of the simulation when the point source is on the ground with the corresponding X-Y coordinate. Similarly, blue and green stems represent the metrics for Z = 1.9 and 3.8 meters respectively. Looking at the results from this scenario, it is evident that whenever a light source is introduced into the environment, which is illuminated only with background light, the performance decreases. This can be inferred from table 2 as well where a comparison between scenario 1 and 4 was made. Table 3 compares scenario 4 against a case from scenario 5 which again bolsters the case that any kind of an illuminating source when introduced into an environment with just background light, deteriorates the performance.

Name of the Scenario	Miss-Count	Distance covered
Scenario 4	70	37.10
Scenario 5	88	36.07

Table 3: Comparison of the results from scenario 4 and 5.

It can be observed from Fig. 18 that the parameter miss count, when the light source is on the ground records a better value than all the other cases and completes the course every time. This adds to the result from scenario 3 where we proved that when a light source is on the ground, the robot tends to follow the human better. The number of miss-counts are almost twice when the light is kept at a height than when it is on the ground. This means that the human following algorithm is performing twice as bad.

These anomalies can be seen when the point light sources are kept at a height from the ground. Looking at Fig. 19, the difference in performance (distance covered) when the light is kept at a height (blue and green stems) can be observed. In most cases, the green and blue stems have only half the amplitude as that of red stems. This means that when the light sources were kept at a height, the robot could follow the human only halfway through. In the simulation, the robot could follow the human only till the point (6, 14) from Fig. 5. This is a difficult turn for the robot to make and when the environment is poorly lit, it finds it even harder to make that turn. The reasons for this behaviour are explained in the discussion section.

VI. Discussion Of Results

We have gathered all our information in the form of tables and graphs, and we have interpreted the results to obtain our inferences. In this section, we would like to discuss and speculate why the robot and hence, the human following algorithm behaves in the way it does in the scenarios listed above.

In scenario 1, The Turtlebot's camera has ample amount of light reflected from the human. This is because the environment is well illuminated by the sun, background and ambient illumination. When there is enough light reflecting from the human, the robot will always be able to detect and follow the human.

Scenario 2 results in the robot not being able to move at all. This is because there is absolutely no light in the environment. As we discussed above, the robot's camera captures the reflected light from the human in order to detect him. But, since there is no light in the environment, there is no scope for light to bounce off the human for the robot to detect and hence the Turtlebot fails to move.

Scenario 3 had two important results. First, When the light source is placed on the ground, the robot's detection and tracking algorithm works better. Second, when the light source is placed behind the human, the robot follows the human better than the cases where the lights were placed in front of the human.

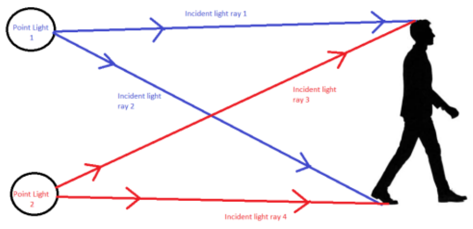

Fig. 20 Incident light from point light sources placed on the ground and at the height of the human.

Fig. 20 shows two of the light rays from each point sources, one on the ground and the other at the height of the human are incident on the human. We can see that each ray of light from a point light source has a counter part in the other point light source which travel the same distance. But this is not the case for reflected lights.

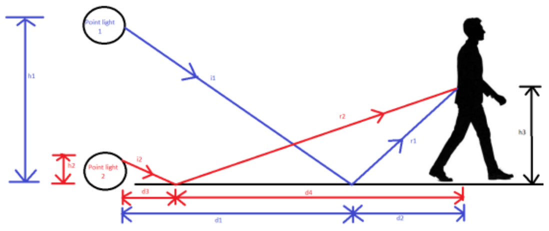

Fig. 21 Light rays from the two-point light sources reflected from the ground and incident on the human

From Fig. 21,

Using Pythagoras theorem,

$$i_1^2 = h_1^2 + d_1^2 \qquad -(1)$$
$$i_2^2 = h_2^2 + d_3^2 \qquad -(2)$$
$$r_1^2 = h_3^2 + d_2^2 \qquad -(3)$$
$$r_2^2 = h_3^2 + d_4^2 \qquad -(4)$$

Adding (1), (3)

$$i_1 + r_1 = \sqrt{(h_1^2 + d_1^2)} + \sqrt{(h_3^2 + d_2^2)} \qquad -(5)$$

Adding (2), (4)

$$i_2 + r_2 = \sqrt{(h_2^2 + d_3^2)} + \sqrt{(h_3^2 + d_4^2)} \qquad -(6)$$

On comparing Equations 5 and, Equation 6, we can see that the ray from point light 1 must travel a larger distance than the ray from point light 2. This is because all the distances on the ground (d1+d2, d3+ d4) add up to the same amount. But since the heights are unequal and h1 > h2, the ray from point light 1 must travel a larger distance than the rays from point light 2. According to wave theory of light, when the light must travel a larger distance, it tends to disperse more and hence the intensity decreases. Since the light from point light 1 travels a farther distance than the light rays from point light 2, it has a comparably lower intensity. The increased intensity of the point light placed on the ground is the reason why the Turtlebot tends to follow the human better when the light source is placed on the ground.

Another important inference from scenario 3 is that whenever the light is placed behind the human, the robot tends to follow the human better. This is because when the light source is placed behind the human, larger amount of light bounces off from the surface of the human. Back of the human is the region of interest for a robot that is trying to follow the human and this scenario well illuminates the human for the sensor to detect and track the human.

Scenario 4 gave the best result so far for a human following robot. This scenario does not incorporate a well-lit environment but involves an environment which has only background light. The reason for this is because of how the algorithm works. The algorithm involves converting the images to binary images for it to function. Binary images are those images which have only black and white components in it. The algorithm converts the images to binary images in the later stages and then predicts on these binary images. Refer to Fig. 22 to understand how the image frames look like from the point of view of the camera on the Turtlebot. This is a very good approximation to how an image converted to binary scale would look like. So, for each cycle, it is easier for the algorithm to detect and classify the image, and hence it speeds up the detection process for the Turtlebot to track and follow the human better.

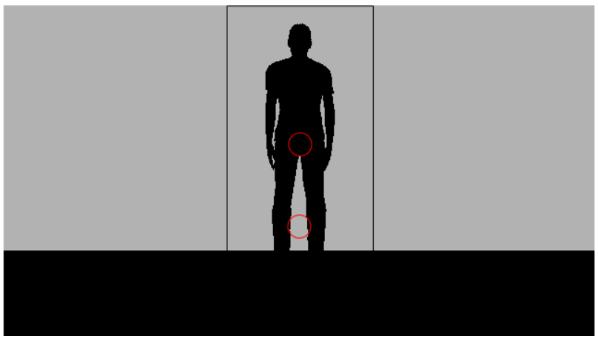

Fig 22. Image frame of the camera on Turtlebot, when the environment has only background light.

Moving on to scenario 5, it was inferred that whenever a light source was placed above the ground in an environment, that was initially illuminated only with background light, the performance degraded. The performance degrades so much that introducing these light sources have a negative effect and thereby inhibits the robot from following the human through the entire trajectory. From the previous scenario it can be seen that whenever the image frames resembled binary images, the detection was better. Also, from scenario 1, whenever the environment was properly lit, the performance was not as good, but the robot was still able to follow the human through the entire trajectory. But when a point light source is introduced at a location and as you move away from the light source the intensity reduces. This means that not enough amount of light is incident on the human but still some amount if it is incident on the human as shown in Fig. 23.

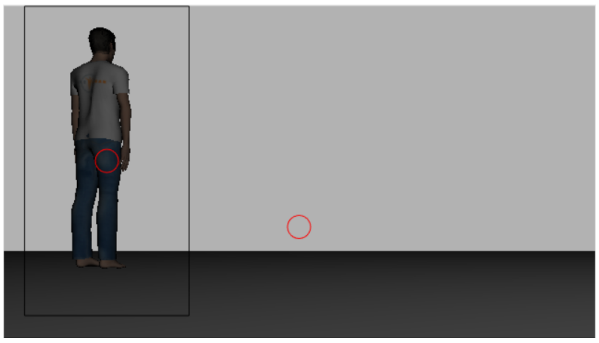

Fig. 23 Image frame when a point light source is present at a distance from the human in an environment with only the background light.

This creates a new problem. Only a part of the human gets illuminated by a very low intensity light and the rest of the human looks like a silhouette. The human is not completely illuminated, and hence the image is neither a binary image nor a well illuminated image of the person, which is ambiguous for the algorithm to detect. This is the reason why the algorithm performs poorly in this scenario.

VII. Future Scope

The proposed simulation experiment can be extended by increasing the variables in the environment. This can be done by adding multiple humans in the scene and increasing the ability of the robot to not just detect a human but to distinguish that human from others in the scene and follow only that person.

Multiple objects can be added to the simulation environment. This can obstruct the view of the bot to find the human and may also block the light from illuminating the human. Robot's performance can be observed in such an environment.

Another interesting idea would be to make the robot human size or even greater. This would make the robot eclipse the human completely when the human, robot and the illuminating source are in a straight line and would make the detection even harder.

The objects added in the scene need not be mere obstacles but other robots in the environment can act as obstacles too. We can have multiple robots following each human uniquely and able to follow any human while avoiding any collision.

The detection and tracking functions of the robot will be vastly affected with the increase in number of variables since it increases the complexity.

We have tested the robot using very specific algorithms and this idea can be extended to others too where the performance of object detection may vary.

The next step would be to fabricate a physical robot and test it in a complicated environment since a computer simulation might not be able to capture all the complexities that would be introduced in the system.

VIII. References

[1] M. Kobilarov, G. Sukhatme, J. Hyams, and P. Batavia, "People tracking and following with mobile robot using an omnidirectional camera and a laser," in *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, 2006, pp. 557–562.

[2] N. Kasundra and K. K. Warhade, "Performance evaluation of object detection and tracking method under illumination variation," in *2014 Annual IEEE India Conference (INDICON)*, 2014, pp. 1–6.

[3] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, 2005, vol. 1, pp. 886–893 vol. 1.

[4] M. Wan, H. Zhang, M. Fu, and W. Zhou, "Motion Control Strategy for Redundant Visual Tracking Mechanism of a Humanoid Robot," in *2016 8th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, 2016, vol. 02, pp. 156–159.

[5] J. Pestana, J. L. Sanchez-Lopez, P. Campoy, and S. Saripalli, "Vision based GPS-denied Object Tracking and following for unmanned aerial vehicles," in *2013 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2013, pp. 1–6.

[6] A. Treptow, G. Cielniak, and T. Duckett, "Real-time people tracking for mobile robots using thermal vision," *Robot. Auton. Syst.*, vol. 54, pp. 729–739, Sep. 2006.

[7] H. Cho, P. E. Rybski, and W. Zhang, "Vision-based bicycle detection and tracking using a deformable part model and an EKF algorithm," in *13th International IEEE Conference on Intelligent Transportation Systems*, 2010, pp. 1875–1880.

[8] Jinyoung Choi, S. Lee, Y. Oh, and S. Oh, "Pedestrian-following service robot applications using chance-constrained target tracking," in *2015 12th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2015, pp. 504–506.

[9] A. A. Godil, R. V. Bostelman, W. P. Shackleford, T. H. Hong, and M. O. Shneier, "Performance Metrics for Evaluating Object and Human Detection and Tracking Systems," *NIST InteragencyInternal Rep. NISTIR - 7972*, Jul. 2014.