Vision Based C3BF in UGV Unicycle Model

Sahil Bobade[1], Anand Sortur[1], Sudharsan Vishwanathan[1], Rushikesh Jadhav[1], Shishir Kolathaya[1]

***Abstract*— This project presents an integration of an object detection algorithm (YOLO V5 / Mask RCNN) with the Collision cone control barrier function (C3BF), to avoid a static obstacle in an environment using Turtlebot3 unicycle model. This integration is facilitated using ROS framework which interacts between the object detection module and the robot's navigation controls. Additionally, the project uses the motion capture system to provide precise positional data of the Turtlebot. The estimation of the depth and size of an obstacle combined with the positional information of bot from the Mocap system enables the C3BF to compute a safe trajectory around an obstacle.**

I. INTRODUCTION

In recent years, the rapid advancement in autonomous robotics has expanded the application of robots across diverse environments. As robots increasingly become part of our everyday environments, they are frequently confronted with static obstacles such as furniture, walls, and other fixed structures. These obstacles demands sophisticated detection and avoidance techniques to ensure safe operation. Also, close interaction with humans demands high reliability in safety-critical tasks. Thus, designing a system which perceives an obstacle in real time and which gives a formal safety guarantees has become an essential aspect of safety-critical applications and an active research area in recent years. For reliable safety measures, such as effective collision avoidance, a control algorithm focused on safety is necessary. This algorithm must be integrated with the trajectory planning system, with a primary emphasis on safety rather than strict adherence to the planned path.

Visual systems provides the bot an ability to perceive the environment in a manner similar to humans. Cameras capture rich, high-resolution data that not only aids in navigation and obstacle avoidance but also in more complex tasks such as object recognition, segmentation, and more. This data richness enhances the ability to make informed decisions based on visual cues. Also, vision-based systems can be adapted and used in a wide range of environments, from indoor settings to rugged outdoor landscapes. Image processing algorithms and machine learning models can be trained to optimize performance under various lighting and weather conditions. A key benefit of employing C3BF-based quadratic programming is its efficiency in real-time applications, allowing for the computation of optimal control inputs at a high frequency. This method can serve as an effective safety overlay on top of existing advanced controllers used for path planning and obstacle avoidance.

In our work, we combine the Visual system with the collision cone control barrier function [1], [2]. In particular, we detect the size and depth of a static obstacle with the camera and also the robot's pose using a motion capture system. These serves as an input to the C3BF controller. The C3BF-based QP method computes the inputs to ensure that the direction of the relative velocity vector remains outside the collision cone (defined as the unsafe set) continuously. This technique has been successfully implemented and demonstrated using an acceleration-controlled unicycle model.

II. BACKGROUND

In this section, we provide the relevant background necessary to formulate the problem of obstacle avoidance for the vehicle that is Unicycle Model . After which, we formally introduce Control Barrier Functions (CBFs) and their importance for real-time safety-critical control.

A. Unicycle Model

The acceleration controlled unicycle model has state variables $x_p, y_p, \theta, v, \omega$ denoting the pose, linear velocity, and angular velocity, respectively. The control inputs are linear acceleration (a) and angular acceleration (α). In Fig. 1 shows unicycle model. The resulting dynamics of this model is shown below:

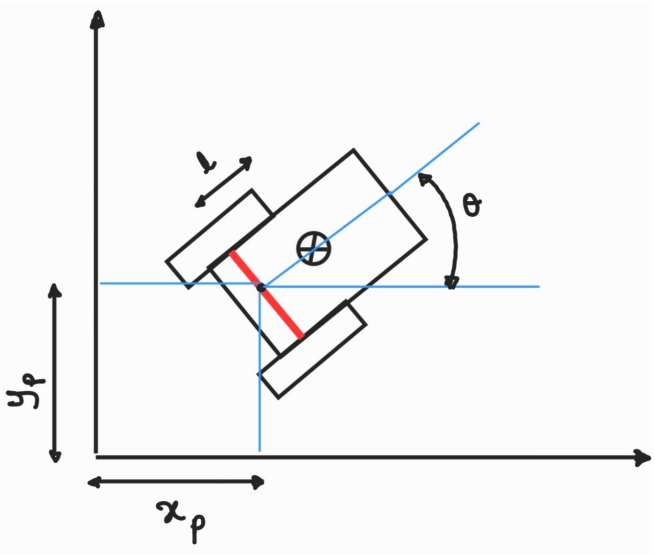

Fig. 1: Obstacle avoidance using visual input

[1]Cyber-Physical Systems, Indian Institute of Science (IISc), Bengaluru.

$$\begin{bmatrix} \dot{x}_p \\ \dot{y}_p \\ \dot{\theta} \\ \dot{v} \\ \dot{\omega} \end{bmatrix} = \begin{bmatrix} v\cos\theta \\ v\sin\theta \\ \omega \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ \alpha \end{bmatrix} \quad (1)$$

In literature we typically see the model involves the linear and angular velocities v, ω as inputs however here we use accelerations as inputs. This is due to the fact that differential drive robots have torques as inputs to the wheels that directly affect acceleration. In other words, we can treat the force/acceleration applied from the wheels as inputs. As a result, v, ω become state variables in our model.

B. Control barrier functions

Having described the unicycle model, we now formally introduce Control Barrier Functions (CBFs) and their applications in the context of safety. Consider a nonlinear control system in affine form:

$$\dot{x} = f(x) + g(x)u \quad (2)$$

where $x \in \mathcal{D} \subseteq \mathbb{R}^n$ is the state of system, and $u \in \mathbb{U} \subseteq \mathbb{R}^m$ the input for the system. Assume that the functions $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^n \to \mathbb{R}^{n\times m}$ are continuously differentiable. Given a Lipschitz continuous control law $u = k(x)$, the resulting closed loop system $\dot{x} = f_{cl}(x) = f(x)+g(x)k(x)$ yields a solution $x(t)$, with initial condition $x(0) = x_0$. Consider a set $\mathcal{C}$ defined as the super-level set of a continuously differentiable function $h : \mathcal{D} \subseteq \mathbb{R}^n \to \mathbb{R}$ yielding,

$$\mathcal{C} = \{x \in \mathcal{D} \subset \mathbb{R}^n : h(x) \geq 0\} \quad (3)$$
$$\partial\mathcal{C} = \{x \in \mathcal{D} \subset \mathbb{R}^n : h(x) = 0\} \quad (4)$$
$$\mathrm{Int}(\mathcal{C}) = \{x \in \mathcal{D} \subset \mathbb{R}^n : h(x) > 0\} \quad (5)$$

It is assumed that $\mathrm{Int}(\mathcal{C})$ is non-empty and $\mathcal{C}$ has no isolated points, i.e. $\mathrm{Int}(\mathcal{C}) \neq \phi$ and $\overline{\mathrm{Int}(\mathcal{C})} = \mathcal{C}$. The system is safe w.r.t. the control law $u = k(x)$ if $\forall x(0) \in \mathcal{C} \implies x(t) \in \mathcal{C} \quad \forall t \geq 0$. We can mathematically verify if the controller $k(x)$ is safeguarding or not by using Control Barrier Functions (CBFs), which is defined next.

Definition 1 (Control barrier function (CBF)): Given the set $\mathcal{C}$ defined by (6)-(8), with $\frac{\partial h}{\partial x}(x) \neq 0 \forall x \in \partial\mathcal{C}$, the function h is called the control barrier function (CBF) defined on the set $\mathcal{D}$, if there exists an extended class $\mathcal{K}$ function κ such that for all $x \in \mathcal{D}$:

$$\underbrace{\sup_{u\in\mathbb{U}}}[\underbrace{\mathcal{L}_f h(x) + \mathcal{L}_g h(x)u}_{\dot{h}(x,u)} + \kappa(h(x))] \geq 0 \quad (6)$$

where $\mathcal{L}_f h(x) = \frac{\partial h}{\partial x} f(x)$ and $\mathcal{L}_g h(x) = \frac{\partial h}{\partial x} g(x)$ are the Lie derivatives.

Given this definition of a CBF, we know that any Lipschitz continuous control law $k(x)$ satisfying the inequality: $\dot{h} + \kappa(h) \geq 0$ ensures safety of $\mathcal{C}$ if $x(0) \in \mathcal{C}$, and asymptotic convergence to $\mathcal{C}$ if $x(0)$ is outside of $\mathcal{C}$.

C. Controller synthesis for real-time safety

Having described the CBF and its associated formal results, we now discuss its Quadratic Programming (QP) formulation. CBFs are typically regarded as safety filters which take the desired input (reference controller input) $u_{ref}(x,t)$ and modify this input in a minimal way:

$$u^*(x,t) = \min_{u\in\mathbb{U}\subseteq\mathbb{R}^m} \|u - u_{ref}(x,t)\|^2 \quad (7)$$
$$\text{s.t. } \mathcal{L}_f h(x) + \mathcal{L}_g h(x)u + \kappa(h(x)) \geq 0$$

This is called the Control Barrier Function based Quadratic Program (CBF-QP).

D. Classical CBFs

Having introduced CBFs, we now explore collision avoidance in unmanned ground vehicles (UGVs). The typical ellipse-CBF Candidate - Unicycle: Consider the following CBF candidate:

$$h(x,t) = \left(\frac{c_x(t) - x_p}{c_1}\right)^2 + \left(\frac{c_y(t) - y_p}{c_2}\right)^2 - 1 \quad (8)$$

which approximates an obstacle with an ellipse with center $(c_x(t), c_y(t))$ and axis lengths c_1, c_2. We assume that $c_x(t), c_y(t)$ are differentiable and their derivatives are piecewise constants. The derivative of (8) is

$$\frac{2\,(c_x - x)\,(\dot{c}_x - v\cos\theta)}{c_1^2} + \frac{2\,(c_y - y)\,(\dot{c}_y - v\sin\theta)}{c_2^2} \quad (9)$$

which has no dependency on the inputs a, α. Hence, h will not be a valid CBF for the acceleration-based model (1). However, for static obstacles, if we choose to use the velocity model (with v, ω as inputs instead of a, α), then h will certainly be a valid CBF, but the vehicle will have limited control capability i.e., it loses steering ω.

E. Collision Cone CBF (C3BF)

Having described the shortcomings of existing approaches for collision avoidance, we will now describe the the collision cone CBFs (C3BFs) [3], [4]. A collision cone, defined for a pair of objects, is a set that can be used to predict the possibility of collision between the two objects based on the direction of their relative velocity. The collision cone of an object pair represents the directions, which if traversed by either object, will result in a collision between the two. For a collision to happen, the relative velocity of the obstacle must be pointing towards the vehicle. Hence, the relative velocity vector must not be pointing into the pink shaded region EHI in Fig. 2, which is a cone. This novel approach of avoiding the pink cone region gives rise to Collision Cone Control Barrier Functions (C3BFs).

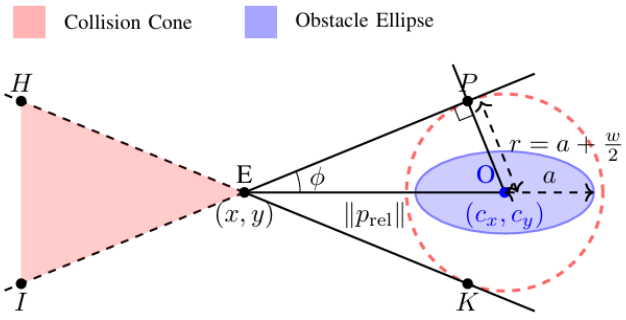

Fig. 2: Construction of collision cone for an elliptical obstacle considering the ego vehicle's dimensions (width: w)

III. Application to Unicycle

Acceleration controlled unicycle model: We first obtain the relative position vector between the body center of the unicycle and the center of the obstacle. Therefore, we have

$$p_{\text{rel}} := \begin{bmatrix} c_x - (x_p + l\cos(\theta)) \\ c_y - (y_p + l\sin(\theta)) \end{bmatrix} \tag{10}$$

Here l is the distance of the body center from the differential drive axis (see Fig. 1). We obtain its velocity as

$$v_{\text{rel}} := \begin{bmatrix} \dot{c}_x - (v\cos(\theta) - l\sin(\theta) * \omega) \\ \dot{c}_y - (v\sin(\theta) + l\cos(\theta) * \omega) \end{bmatrix} \tag{11}$$

We propose the following CBF candidate:

$$h(x,t) = < p_{\text{rel}}\,, v_{\text{rel}} > + \|p_{\text{rel}}\|\,\|v_{\text{rel}}\|\cos\phi \tag{12}$$

where, ϕ is the half angle of the cone, the expression of $\cos\phi$ is given by $\frac{\sqrt{\|p_{\text{rel}}\|^2 - r^2}}{\|p_{\text{rel}}\|}$ (see Fig. 2). The constraint simply ensures that the angle between $p_{\text{rel}}\,, v_{\text{rel}}$ is less than $180° - \phi$. We have the following first result of the paper:
Theorem 1: Given the acceleration controlled unicycle model (1), the proposed CBF candidate (12) with $p_{\text{rel}}, v_{\text{rel}}$ defined by (10), (11) is a valid CBF defined for the set $\mathcal{D}$.
Proof: Taking the derivative of (12) yields $(v_{\text{rel}} \neq 0)$

$$\begin{aligned} \dot{h} = &< \dot{p}_{\text{rel}}\,, v_{\text{rel}} > + < p_{\text{rel}}\,, \dot{v}_{\text{rel}} > \\ &+ < v_{\text{rel}}\,, \dot{v}_{\text{rel}} > \frac{\sqrt{\|p_{\text{rel}}\|^2 - r^2}}{\|v_{\text{rel}}\|} \\ &+ < p_{\text{rel}}\,, \dot{p}_{\text{rel}} > \frac{\|v_{\text{rel}}\|}{\sqrt{\|p_{\text{rel}}\|^2 - r^2}} \end{aligned} \tag{13}$$

Further $\dot{p}_{\text{rel}} = v_{\text{rel}}$ and

$$\dot{v}_{\text{rel}} = \begin{bmatrix} -a\cos\theta + v(\sin\theta)\omega + l(\cos\theta)\omega^2 + l(\sin\theta)\alpha \\ -a\sin\theta - v(\cos\theta)\omega + l(\sin\theta)\omega^2 - l(\cos\theta)\alpha \end{bmatrix}$$

Given $\dot{v}_{\text{rel}}$ and $\dot{h}$, we have the following expression for $\mathcal{L}_g h$:

$$\mathcal{L}_g h = \begin{bmatrix} < p_{\text{rel}} + v_{\text{rel}} \frac{\sqrt{\|p_{\text{rel}}\|^2 - r^2}}{\|v_{\text{rel}}\|}, \begin{bmatrix} -\cos\theta \\ -\sin\theta \end{bmatrix} > \\ < p_{\text{rel}} + v_{\text{rel}} \frac{\sqrt{\|p_{\text{rel}}\|^2 - r^2}}{\|v_{\text{rel}}\|}, \begin{bmatrix} l\sin\theta \\ -l\cos\theta \end{bmatrix} > \end{bmatrix}^T$$

It can be verified that for $\mathcal{L}_g h$ cannot be zero as proved in this paper [1],

A. Methodology

The Intel RealSense D435i camera was mounted on the TurtleBot (fig 4). It offers high-resolution depth sensing by combining stereo vision with infrared depth sensing. The depth frame rate is 90 fps and the RGB frame rate is 30 fps. The RGB field of view (H × V) is 69° × 42°

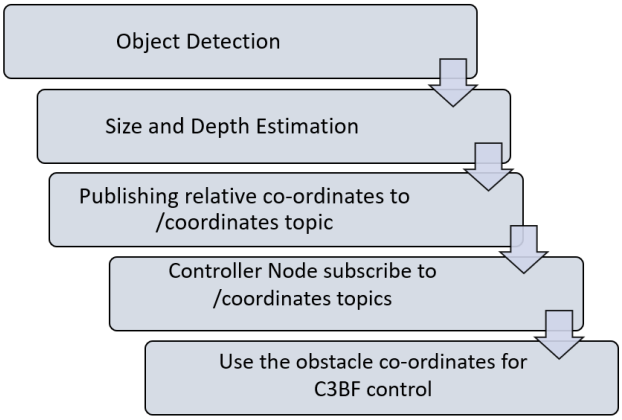

Fig. 3: Flow of operation

Fig. 4: Turtlebot with camera and MoCap sensor

Using the depth stream from the RealSense camera, the depth at the centroid of the detected object's bounding box is extracted. This depth information, coupled with the camera's field of view and the object's pixel dimensions, is used to

calculate the real-world size and also the coordinates of an object.

The object is detected using the two methods:

1. Mask RCNN
2. YOLO V5

B. Mask RCNN

Mask R-CNN is a deep learning model for object detection and instance segmentation. It extends Faster R-CNN by adding a branch for predicting segmentation masks on each Region of Interest (RoI), independent of the class box. It identifies objects and their contours within an image, providing pixel-level precision which is crucial for size measurement. This is particularly valuable for detailed environment mapping and interaction, allowing for precise obstacle avoidance strategies. The number of parameters for this model are 43.5 million.

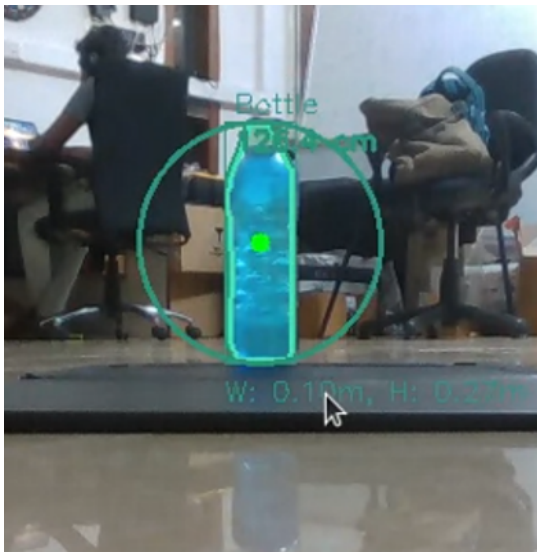

Fig. 5: Mask RCNN Object size detection

The architecture of Mask R-CNN is as follows:

Region Proposal Network (RPN): In the first stage, the model generates a set of region proposals that are likely to contain objects. These proposals are potential bounding boxes around objects in the image. The region proposal network is responsible for suggesting these candidate regions.

Region of Interest (RoI) Pooling: Once the region proposals are generated, each region is cropped from the image and resized to a fixed size. This process is known as RoI pooling, and it ensures that the region of interest is consistently represented in a fixed-size feature map, regardless of the size or aspect ratio of the original region proposal.

Feature Extraction: The cropped and resized regions are then passed through a pre-trained convolutional neural network (CNN) to extract features from each region.

Object Classification and Bounding Box Regression: The features extracted from each region are used for two tasks: object classification and bounding box regression. Object classification involves determining the class of the object within the region, and bounding box regression refines the coordinates of the bounding box around the object.

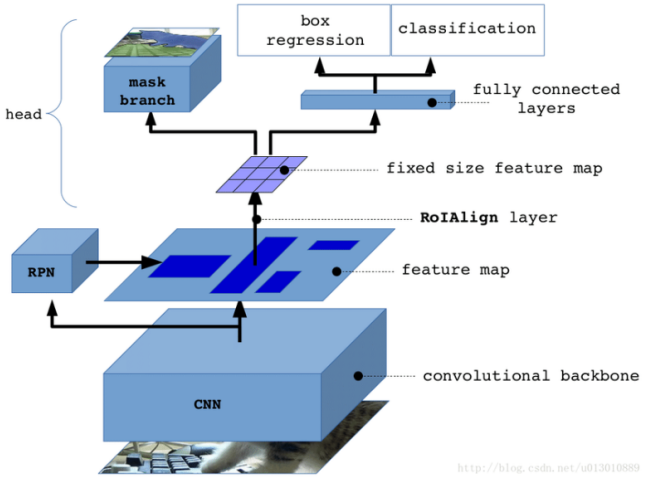

Fig. 6: Mask RCNN architecture

C. YOLO V5

YOLOv5 is a state-of-the-art deep learning model designed for real-time object detection tasks, making it well-suited for the project's objectives. YOLOv5 offers high accuracy in object detection, providing TurtleBot with precise information about the location and dimensions of static obstacles. This accuracy is essential for calculating safe trajectories and implementing effective obstacle avoidance strategies. There are 5 versions of YOLOv5, namely YOLOv5x, YOLOv5l, YOLOv5m, YOLOv5s, and YOLOv5n. We are using YOLOV5s i.e small version.

YOLOv5's architecture consists of three main parts:

Backbone: This is the main body of the network. For YOLOv5, the backbone is designed using the New CSP-Darknet53 structure, a modification of the Darknet architecture.

Neck: This part connects the backbone and the head. In YOLOv5, the SPPF and New CSP-PAN structures are utilized.

Head: This part is responsible for generating the final output. YOLOv5 uses the YOLOv3 Head for this purpose.

Its lightweight architecture ensures minimal computational overhead, enabling real-time performance without sacrificing accuracy.

In the Fig we can see that YOLO V5 is able to detect the size and the depth of an obstacle.

D. Object Size calculation

The obstacle size calculation for an image of 640*480 is carried out as follows.

$$\text{width} = 2 \cdot \left(depth \cdot \tan\left(\frac{\text{fov\_x}}{2} \right) \right) \cdot \frac{\text{width\_px}}{640}$$

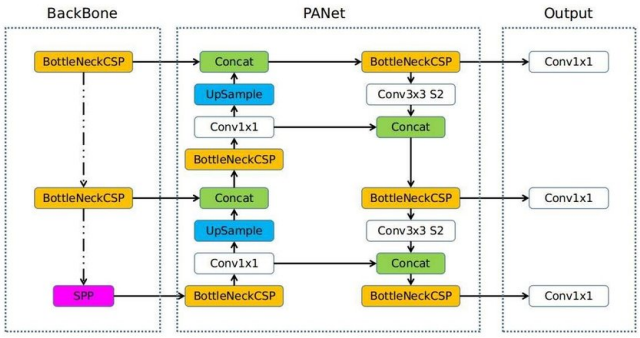

Fig. 7: YOLO V5 architecture

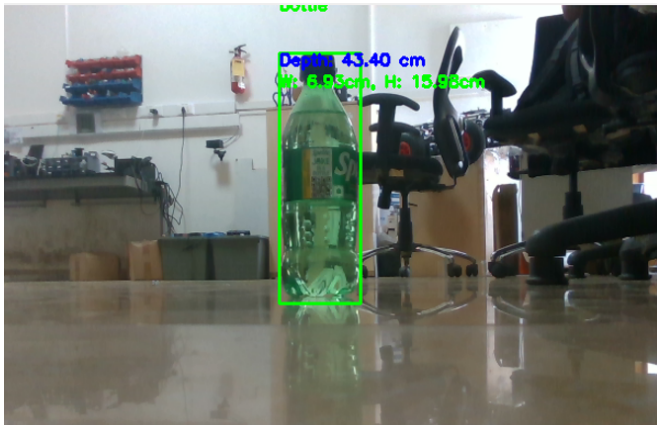

Fig. 8: YOLO V5 object depth and size estimation

$$\text{height} = 2 \cdot \left(depth \cdot \tan\left(\frac{\text{fov\_y}}{2} \right) \right) \cdot \frac{\text{height\_px}}{480}$$

where fov_x and fov_y are the Horizontal field of view and the Vertical field of view respectively.

fov_x = 69.4

fov_y = 42.5

width_px and height_px are the Width and Height in pixels respectively.

Also, the offset of an obstacle from the center of a turtlebot is calculated to check whether the obstacle is in the path of the bot or not.

In this way, we get the size, depth, and co-ordinates of an obstacle in robot frame. This information is published in to ROS topic (/coordinates). Controller node subscribe to this topic and compute global co-ordinates of an obstacle. This information is used in the computation of the control barrier function.The state information for turtlebot3 is being published by the Mocap phase marker ROS topic.

IV. Results and Discussions

In this section, we provide the results to validate the Vision based C3BF. All the tests are done using TurtleBot3 (modeled as a unicycle) and using a Motion capture system.

A. Static obstacle avoidance using Camera

We have considered the reference control inputs as a simple PD controller. Constant target velocities were chosen

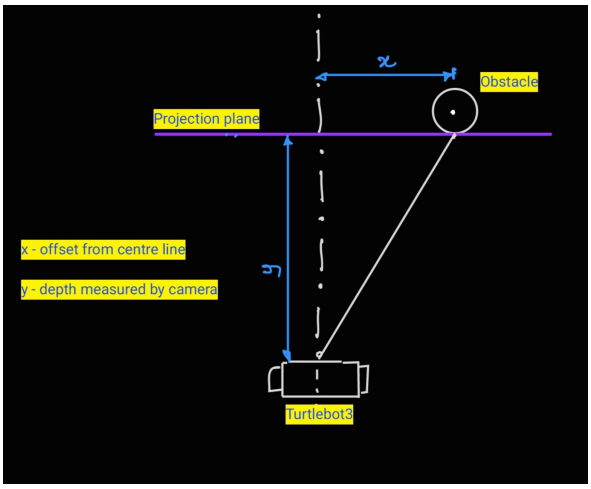

Fig. 9: Offset calculation

to verify the C3BF-QP. For the class $\mathcal{K}$ function in the CBF inequality, we chose $\kappa(h) = \gamma h$, where $\gamma = 1$ and the weight matrix for linear acceleration was considered as 10 times the weight of angular acceleration. This was done to ensure that Turtlebot meets the constant target velocity of 0.5 m/s while avoiding the obstacle. The width of the obstacle obtained from vision output was accounted while formulating the C3BF. Fig. shows turlebot avoiding a static obstacle using visual input.

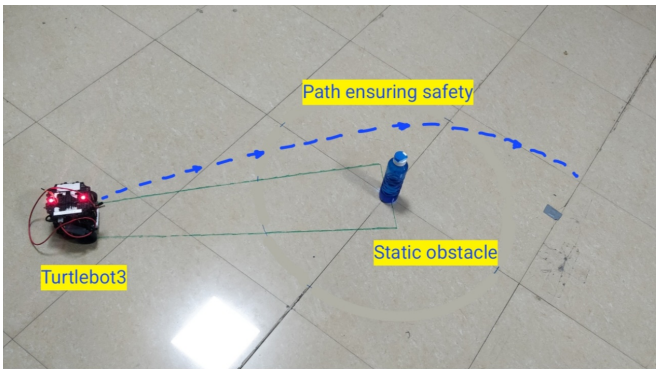

Fig. 10: Obstacle avoidance using visual input

B. Static obstacle fed as prior information

The coordinates of the obstacle in the world frame are pre-fed as input for the C3BF formulation. Here the width of the obstacle was assumed to be uniform with a radius of 50cm. Here the weight matrix for linear acceleration was considered as 100 times the weight of angular acceleration. This was done to ensure the Turtlebot meets the constant demand velocity of 0.5 m/s while avoiding the obstacle. We have considered the reference control inputs as a simple PD controller. For the class $\mathcal{K}$ function in the CBF inequality, we chose $\kappa(h) = \gamma h$, where $\gamma = 1$. Fig. shows turlebot avoiding an obstacle whose information is prefed in Mocap

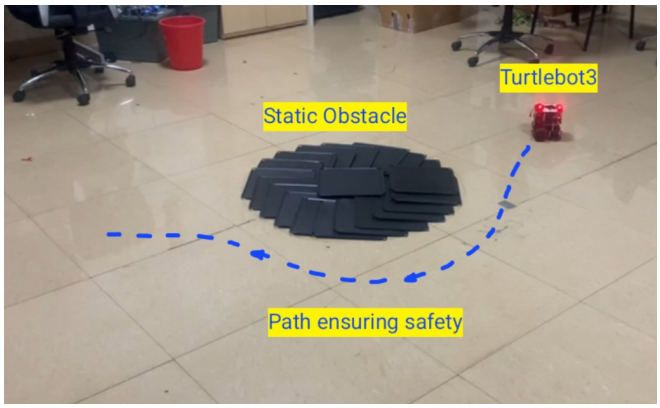

Fig. 11: Obstacle avoidance by prefed Mocap

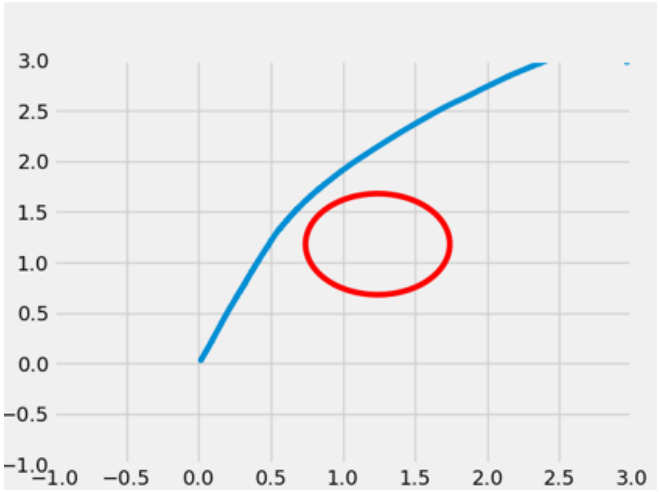

Fig. 12: Trajectoryof Obstacle avoidance by prefed Mocap info

C. Dynamic obstacle avoidance using MoCap

In dynamic obstacle, omnibot was considered as a moving obstacle. The position and the velocity of the obstacle was computed using the phase space marker information as published over ROS topic. The omnibot was moving in trajectory lane which was intersecting with the Egobot trajectory. It was controlled using PD law without using any control barrier function. The linear and angular acceleration of the Egobot was computed taking into account the state information of the omnibot (obstacle). Blue is ego vehicle and red is omnibot.

V. Conclusions

We demonstrated the use of vision in estimating the size and the depth of a static obstacle. These size and depth measurements will be input to the collision cone control barrier function controller, thus allowing the turtlebot unicycle model to safely navigate through an environment with a static obstacle.

VI. Acknowledgement

We would like to thank Karthik, Tejas, Shubham, Shishir Sir and Pushpak Sir. Special Thanks to Manan for helping throughout the project.

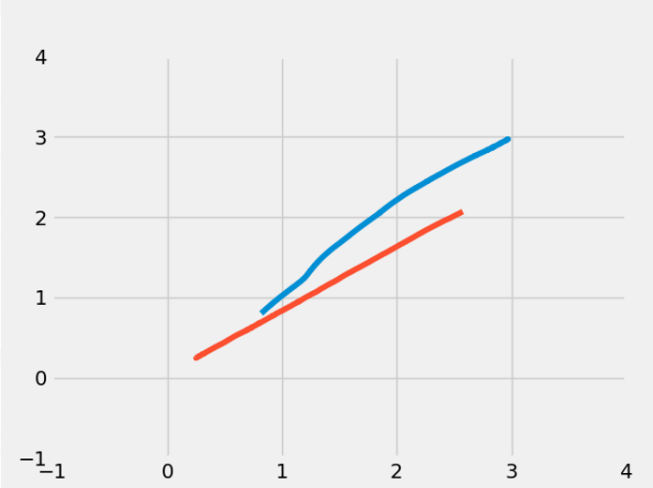

Fig. 13: .Dynamic obstacle avoidance using MoCap info

VII. Future Scope

Avoiding dynamic obstacles using vision is the obvious next step of this project. It will require us to estimate the state of a dynamic obstacle in real time. Also, Integration of multiple sensors, such as cameras and Lidar through fusion techniques can significantly improve detection accuracy and robustness. Thus, we aim to enhance the autonomy and safety of autonomous vehicles by enabling them to avoid static as well as dynamic obstacle in diverse environments.

Moreover, we would like to extend it to aerial vehicles [3], legged robots [5], [6] in cluttered environments [7].

References

[1] M. Tayal, B. G. Goswami, K. Rajgopal, R. Singh, T. Rao, J. Keshavan, P. Jagtap, and S. Kolathaya, "A collision cone approach for control barrier functions," *arXiv preprint arXiv:2403.07043*, 2024.

[2] B. G. Goswami, M. Tayal, K. Rajgopal, P. Jagtap, and S. Kolathaya, "Collision cone control barrier functions: Experimental validation on ugvs for kinematic obstacle avoidance," *arXiv preprint arXiv:2310.10839*, 2023.

[3] M. Tayal and S. Kolathaya, "Control barrier functions in dynamic uavs for kinematic obstacle avoidance: a collision cone approach," *arXiv preprint arXiv:2303.15871*, 2023.

[4] P. Thontepu, B. G. Goswami, M. Tayal, N. Singh, S. S. P I, S. S. M G, S. Sundaram, V. Katewa, and S. Kolathaya, "Collision cone control barrier functions for kinematic obstacle avoidance in ugvs," in *2023 Ninth Indian Control Conference (ICC)*, 2023, pp. 293–298.

[5] M. Tayal and S. Kolathaya, "Safe legged locomotion using collision cone control barrier functions (c3bfs)," *arXiv preprint arXiv:2309.01898*, 2023.

[6] G. Mothish, K. Rajgopal, R. Kola, M. Tayal, and S. Kolathaya, "Stoch biro: Design and control of a low cost bipedal robot," *arXiv preprint arXiv:2312.06512*, 2023.

[7] M. Tayal and S. Kolathaya, "Polygonal cone control barrier functions (polyc2bf) for safe navigation in cluttered environments," *arXiv preprint arXiv:2311.08787*, 2023.