# Object Tracking for Safety

Saptadeep Debnath, Preeti Kannapan, Malvika Shetty
College of Engineering, University of Michigan, Ann Arbor, MI, USA
{saptadeb,preetimk,malvikad}@umich.edu

## I. EXECUTIVE OVERVIEW

Over the past few years, there has been tremendous research on the topic of object tracking. The main aim of our project is to develop a safety warning system by implementing multiple object tracking in videos using bounding boxes. We rely on YOLO, a state-of-the-art object detection model, to detect objects in the image frame and the DeepSORT algorithm for tracking the detected objects. Using an RGBD dataset from the Princeton Tracking Benchmark [6], we transformed the position coordinates of the tracked objects from the image frame $(u, v, d)$ to the camera frame $(x, y, z)$, and calculated their corresponding velocity vectors. These position and velocity vectors were then used to develop a safety warning system which takes into consideration the distance and velocity at which the objects are approaching each other.

## II. BACKGROUND AND IMPACT

With more and more autonomous robots and vehicles being deployed in the world, it is becoming increasingly important to ensure the safety of everyone in the robot's environment. There are many contexts in which this is applicable – i) human-robot interactions e.g. collaborative robots, service robots ii) robot-robot interactions e.g. industrial robots iii) robot-object interactions e.g. obstacle avoidance in autonomous vehicles. Such environments are highly dynamic in nature and can easily lead to hazardous situations for the humans present. While most robots are equipped with IR sensors to perform tasks in their vicinity safely, it still leaves them with the disadvantage of being near-sighted. As we have learnt, vision is hard – while humans can gauge danger instinctively, it is much harder to have a robot do the same. With such robots being deployed autonomously in human-interactive environments, a smart safety system which can issue warnings before possible collisions by simply taking in an RGBD video stream would be very valuable.

There are a variety of ways in which this project could positively impact society. One way in which our safety system can be used is as an external device that monitors a scene, and issues warnings before possible collisions between objects (which could be robots, humans etc.) in the scene. In this case, the system has the advantage of a having a large field of view, and would be beneficial in many situations: i) Industrial environments often have heavy autonomous robots, large moving objects, as well as human workers. A collision between any of these could be disastrous to humans, and safety warnings would help avoid this. ii) With the ongoing pandemic this year, there has been a large focus on contact-less service robots in hospitals, restaurants etc. Such environments with moving robots, humans and objects could also be monitored by our system. A second way in which the safety warning system can potentially be used is as a mounted device on an object in a dynamic environment. Autonomous vehicles integrated with this system could be made much safer, as it makes use of object tracking. For example, based on an object's current trajectory, the system could possibly predict collisions even if the object is not directly headed towards the vehicle. The available position and velocity information could also be used to have the vehicle react appropriately, i.e. move in the right direction with enough speed to avoid collision. Therefore, there is a lot of scope for this system to be used as it is, and also to be extended further.

## III. METHOD

### A. YOLO Object Detection

This project uses the state of the art single shot object detection method YOLOv3 [1]. First, the algorithm tries to predict the bounding box parameters ($b_x, b_y, b_w$ and $b_h$) using dimension clusters as anchor boxes [2] for a given ground truth object which further utilises a sigmoid function. A class label is then predicted for each bounding box at three different scales. Binary entropy loss is used during training for class predictions. A 53 layer convolutional network is implemented for feature extraction. The weights computed on the COCO dataset [3] are used for this project.

EECS 504 Team SafeBot Final Project Report
[1]Presentation Video:
https://youtu.be/aILSsw7K2z8
[2]Project website:
https://eecs504-f20.github.io/object-tracking-for-safety/
[3]Git Repository:
https://github.com/eecs504-f20/object-tracking-for-safety

### B. DeepSORT

It is critical for the project to lock on to the detecting object and issue a warning if the objects approach each other. For the same purpose a mere object detection system is not enough and requires a framework where the objects are detected and are kept track of throughout the entire video. There are a few traditional methods which are more commonly used, such as Mean Shift, Optical Flow and Kalman Filter. Some of the drawbacks of using these include - computational complexity, being prone to noise, data with occlusion.

Hence, a modified DeepSORT [4] (Simple Online Real-time Tracking [5]) algorithm is used for object tracking. This uses a Kalman filter which is robust to noise in the data and predicts the centres of each of the bounding boxes and the corresponding velocities in the concurrent frames. This information is crucial to tracking a given bounding box in to simultaneous frames. The Mahalanobis distance metric is used to incorporate the uncertainties from the Kalman filter. This method also incorporates the object coming in and going out of the image frame. New boxes are introduced with zero velocities for the newly introduced objects in the frame. It is also found through experimentation that it keeps a record of the previously identified objects and is able to re-label and re-track the object if it appears in the frame at a certain point. A distance metric based on the appearance of the object is the deep learning aspect of the algorithm. This addition reduces the number of identity switches and increases efficiency due to occlusion. For our implementation in this project a pre-trained DNN model is used for DeepSORT.

### C. Safety Module

The safety warning system developed in this project utilizes the positions and velocities of the tracked objects with respect to the camera frame. The camera frame was chosen because in this project, we focus on detecting collisions between objects within the scene of interest. However, this method can also be extended to detect collisions between objects and the camera itself. For simplicity, the objects have been assumed to be point masses at the centroids of their bounding boxes.

Let the centroid coordinates in the image frame be denoted by $(u, v)$ and the corresponding raw depth value by $d$. This raw depth value is transformed to camera coordinates ($z$) via a bit-shifting process. The centroid coordinates are then transformed from the image frame $(u, v)$ to the camera frame $(x_{cam}, y_{cam})$ using the following camera intrinsic parameters: the principle point image coordinates $(c_x, x_y)$ and the focal lengths in the $x$ and $y$ directions $(f_x, f_y)$ -

$$x_{cam} = (u - c_x) \cdot z/fx;\ y_{cam} = (v - c_y) \cdot z/fy;\ z_{cam} = z \tag{1}$$

The velocities of the objects with respect to the camera frame are then calculated between the current and previous video frames. Each component of the velocity vector $v_m$ ($m \in x, y, z$) for an object $j$ is given by -

$$V_m = (m_{cam}[i, j] - m_{cam}[i - 1, j]) \cdot fps \tag{2}$$

Here, $i$ denotes the frame number, and $fps$ is the frame rate.
The safety warning system uses the positions and velocities of the tracked objects to evaluate certain safety conditions. Three situations that warrant a warning have been identified as follows:

1) Object $i$ approaches object $j$ at a fast pace from far away
2) Object $i$ approaches object $j$ at a slow pace from close-by
3) Object $i$ approaches object $j$ at a fast pace from close-by

The first condition is used to issue the warning "APPROACHING", while the second and third conditions are used together with an 'or' to issue the warning "DANGER".

To formalize the conditions mentioned above, the relative distance and velocity of the objects are required: $p_{ij} = p_i - p_j$; $V_{ji} = V_j - V_i$ Let $V_{approach_{i,j}} = p_{ij} \cdot V_{ji}$, where the dot indicates the vector inner product. When $V_{approach_{i,j}}$ is positive, we know that the two objects are approaching each other. This can be understood as follows: if the velocity of object A with respect to object B ($V_{AB}$) has a positive component ($V_{approach_{B,A}}$) along the line joining A and B in the direction of B ($p_{BA}$), then the distance between the objects is reducing at the rate of $V_{approach_{B,A}}$. Then, the above conditions can be written mathematically as follows:
Condition 1: Far away but approaching fast

$$p_{ij} > distance_{tol} \text{ and } V_{approach_{i,j}} > approaching_{tol}$$

Condition 2: Close-by but approaching slow

$$p_{ij} \leq distance_{tol} \text{ and } 0 < V_{approach_{i,j}} \leq approaching_{tol}$$

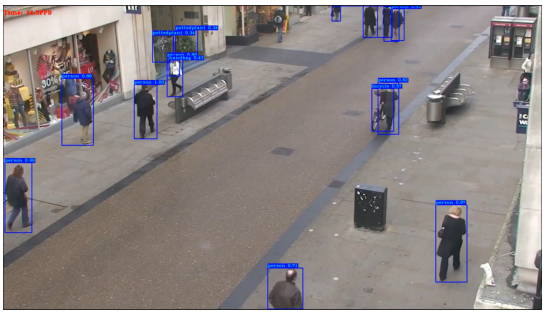
Condition 3: Close-by and approaching fast

$$p_{ij} > distance_{tol} \text{ and } V_{approach_{i,j}} > approaching_{tol}$$

Here, $distance_{tol}$ and $approaching_{tol}$ are tolerance values than can be tuned.

## IV. PROTOTYPE

To implement this prototype, the YOLO toolkit was first used to obtain object detections. It was tested on an image of a busy street, and detections of people, cars, etc. were obtained as bounding boxes and labels on the image (Fig. 1a). Next, YOLO object detection with DeepSORT tracking was tested on a sample video and detections along with tracking IDs were obtained for each video frame (Fig. 1b).



(a) YOLOv3 multilabel-multiclass implementation on a video of the street

(b) DeepSORT implementation using bounding box detections from YOLO and filtered for 'Persons' class.

Fig. 1: Computer vision techniques used for this project.

Once we had the YOLO toolkit working with DeepSORT for object tracking, we chose an RGBD video sequence dataset from the Princeton Tracking Benchmark called "Three People". The dataset contains sequences of RGB frames and depth frames for a video of three people moving about a room. The RGB frame sequence was first converted into an RGB video by calculating the frame rate from the frame time stamps. This video was then fed into the tracking function, and an RGB video with the three people tracked was obtained (Fig. 2a).

In order to check for safety, the positions and velocities of the tracked objects need to be extracted in each frame. To do this, the centroid of each bounding box was calculated using the given corner coordinates, width and height $(x, y, w, h)$: $u = x + w/2, v = y + h/2$. The raw depth value $d$ was then extracted at $(u, v)$ from the corresponding depth frame, and transformed to camera coordinates $z_cam$ as described previously. Next, the centroid coordinates $(u, v)$ of each object were transformed to camera coordinates $(x_{cam}, y_{cam})$ using eqn. 1. Then, the velocity of each object between the current and previous frame ($V_x$, $V_y$, $V_z$) was calculated using eqn. 2.

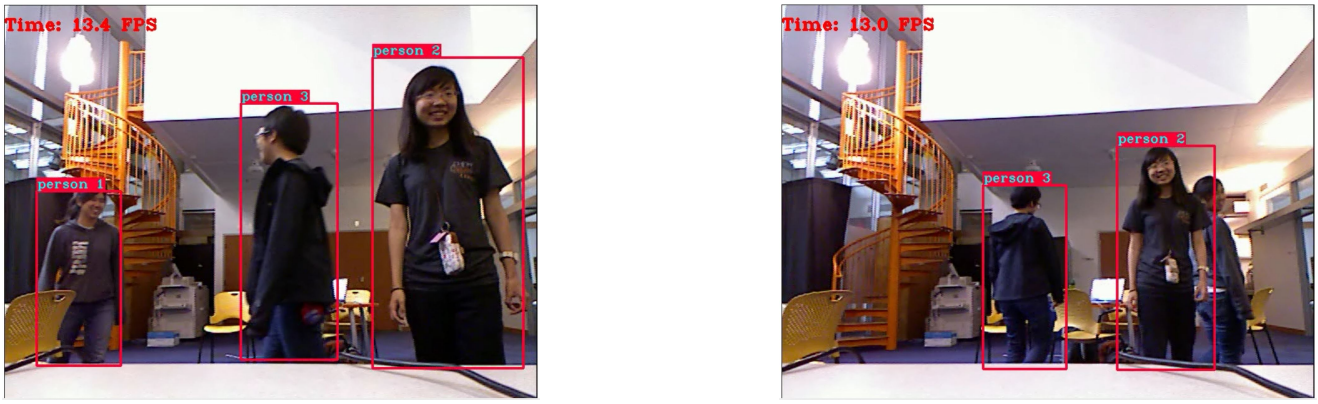
We then projected these position and velocity vectors onto the $x - z$ plane to get a 2D top view of the scene for better visualization (Fig. 3a). For each pair of objects in the scene the distance between them ($p_{ij}$) and their relative velocity ($v_{ij}$) was calculated. This was used to evaluate the safety conditions described previously, and issue the appropriate warning (Fig. 3b). The tuned tolerance values used were $distance_{tol} = 0.3m$ and $approaching_{tol} = 0.5m/s$. We found that for our chosen video, the second condition caused the system to err too much on the side of safety, since all the velocities are quite low. We removed this condition in our current implementation, but these conditions and the corresponding tolerances can be chosen appropriately depending on the situation.

An interesting problem that we ran into was that of missing position and velocity information during occlusion. When one of the people moved behind another person, the bounding box tracking the hidden person naturally disappeared and only reappeared when enough of the hidden person was back in sight. In these frames, we did not have bounding box centroid or depth information for the hidden person (Fig. 2b). Hence, we also had missing position and velocity values in these frames. To overcome this problem, we simply fixed the position and velocity to be that of the last frame in which the bounding box was present. This did result in slight jumps in position before and after the occluded frames, as well as temporarily outdated velocity vectors. However, this has not affected the performance of the system very adversely. In fact, in a particular frame where one person moves backward towards a person hidden behind them, the system still correctly issues a warning. This has been achieved by tuning the safety thresholds such that the system picks up on possibly approaching collisions even if the positions and velocities are slightly erroneous.

## V. RESULTS

In this section, we will be discussing the performance of our implementation of the object tracking module, safety module and final safety warning system.

We were successfully able to use the YOLO toolkit on the RGB video sequence of our chosen Princeton RGBD dataset "Three People" to obtain detections of the 'Person' class. The algorithm is able to detect persons even when they are partially occluded - as can be seen in Fig. 2a, Person 1 is detected even though she is partially hidden by a chair. However, it faces an issue when more than half of the person is occluded by another object in the camera view (Fig. 2b). The DeepSORT algorithm performs well on our dataset in spite of this issue; tracking each person with the correct ID even if they were occluded in previous frames.



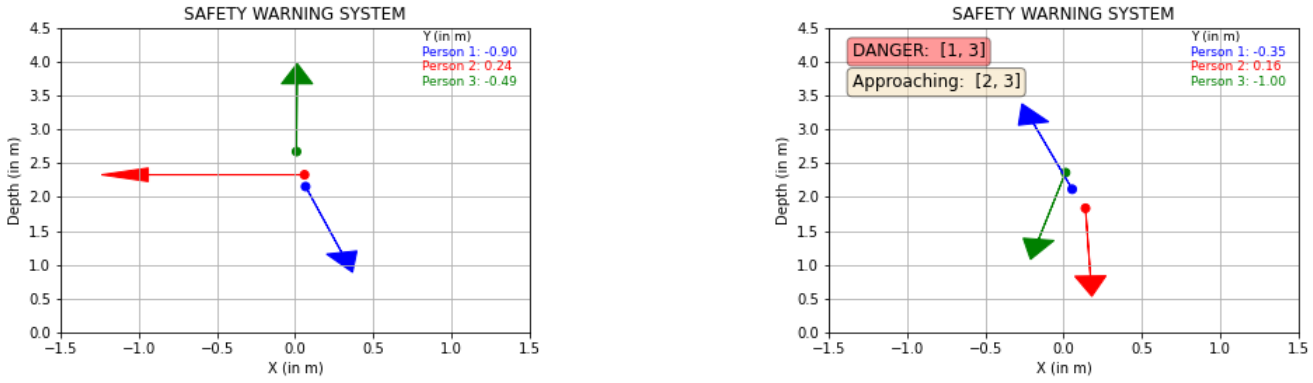

(a) Three People Tracking

(b) Occlusion Problem

Fig. 2: Object Tracking using YOLO and DeepSORT

With the three persons tracked accurately in the RGB video sequence, we were successfully able to integrate this with the depth video sequence to generate a 2D top view of their positions as dots and velocity vectors as arrows with respect to the camera frame (Fig. 3a). The 2D view is our safety module interface and it shows the positions of the persons in terms of their $x_{cam}$ coordinate and depth coordinate $z_{cam}$.

Taking into consideration the relative distance and velocity between the persons, our safety warning system is successfully able to issue an appropriate warning (APPROACHING or DANGER) based on the situation. Since the warning system is based on relative positions and velocities, even if a velocity vector is currently pointing at another person's position, it will not issue a warning if the person is moving away fast enough. Similarly, even if a velocity vector is not pointing directly at another person (Fig. 3b), the system still issues a warning (in this case, APPROACHING), since the distance between them is reducing. Thus, the results of our safety warning system have been verified to be performing accurately, both with respect to the 2D view as well as when compared to the corresponding object tracking video.




(a) 2D top view

(b) Safety Warnings

Fig. 3: Safety Warning System

## REFERENCES


[1] Redmon, J. and Farhadi, A., 2018. *Yolov3: An incremental improvement.* arXiv preprint arXiv:1804.02767.

[2] Redmon, J. and Farhadi, A., 2017. *Yolo9000: Better, faster, stronger.* In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 7263-7271).

[3] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P. and Zitnick, C.L., 2014, September. *Microsoft coco: Common objects in context.* In European conference on computer vision (pp. 740-755). Springer, Cham.

[4] Wojke, N. and Bewley, A., 2018, March. *Deep cosine metric learning for person re-identification.* In 2018 IEEE winter conference on applications of computer vision (WACV) (pp. 748-756). IEEE.

[5] Wojke, N., Bewley, A. and Paulus, D., 2017, September. *Simple online and realtime tracking with a deep association metric.* In 2017 IEEE international conference on image processing (ICIP) (pp. 3645-3649). IEEE.

[6] *Princeton Tracking Benchmark, http://tracking.cs.princeton.edu/dataset.html*