Escape Router

Prithvi Dev Kottimukkaloor Venkatraman[1], Thivish Naran Rajaganesh[2]

***Abstract*— Fire accidents, being one of the most occurred accidents (on average, there are more than 13 million fires in the United States each year, causing over 3,300 deaths and 14,000 injuries annually). Recent studies suggest that the probability of aged people dying in a fire accident (ages 65-74 had 2.2 times the risk of dying in a fire and the 10-year (2011-2020) fire death rate trend for this age group increased 22%) is more nowadays. Many firefighting personnel lost their lives while saving the lives of many. Therefore, we have arrived at a solution to lower those rates at least. Our objective is to monitor the fire, predicting the fire spread direction in an open auditorium/hall, and indicating us the optimal exit path for the people to exit safely and also for the FD officers to take the best exit to carry out their rescue operations.**

I. Introduction

All buildings have a number of different exit paths. But being able to choose the optimal one will always be the most crucial step during times of emergency. Many factors affect this decision to choose like the shortest distance or a wheelchair accessible one or the safest one as well.

We have planned on creating an algorithm to choose the optimal exit path based on the safety factor. We have scaled down the experimental setup to determine the escape path inside a large room with multiple exits. We will be training and validating our model with various data sets of fire spreading in different directions so that it can be trained to be robust in detecting the safest exit path.

There are 2 main tasks to be done to get the safest exit - trajectory coordinate prediction and the exits coordinates or its locations in the ptcloud.

For the first task, we detect the fire using the HSV method and using the bounding box method, got the fire spread trajectory coordinates for all data sets. We are also training a model using linear regression concept, with the trajectories dataset as input, to predict the possible firespread coordinates for a given trajectory.

A random point cloud is created by keeping this fire spread coordinates as its centroid and have assumed this created point cloud to be the fire as we had detected fire using a 2D camera.

For the second task, using our own created pointcloud of an apartment basement, we try to find the exits coordinates in it by using RASNAC to fit the optimal plane containg exits pointclouds and the using KDTree to find the Nearest Neighboring Points, so that the individual points of the exits can be retrived.

Then combining the two, using the predicted firespread coordinates and known exit coordinates, we developed an algorithm which computes the longest distances from the created fire point cloud so that it will the safest exit out of all and draw a line trajectory from point cloud origin or the place were we assumed the people are located, towards the safest exit

[1]New York University, Brooklyn, NY 11201, USA {author1, author2}@nyu.edu

II. Related Work

There are various methods that were employed previously to detect fire like training and validating an algorithm to detect fire or deep learning [1] or using predefined object detection functions [2] or using geometric active contours [3]. One of the most challenging aspects of these methods is the need for large data sets for accurately measuring the fire contours. We would need to collect large data sets and the trained model would have a high accuracy rate.

But for our project, we would not need such a high degree of accuracy as data collection as well as training the model takes a significant amount of time, rather we would want the center point of the detected fire to track its trajectory. Since the application of our projects just needs the coordinates of the fire we use the method of Hue Saturation Value such as the work by Zarkasi et al. [4]. Here we would detect the fire based on hue, saturation and values of the fire color to be detected and get the coordinates of the fire.

Next coming to the fire spread trajectory coordinates, it can be predicted in various ways such as the work by Khennou et al. [5], where the author has used deep learning to train the model by taking into external factors also into consideration. When the nearest neighbour approach is done for computing neighbour points it is noted that the points are calculated both along the axis and the other axis that noted from the work by Calvoa et al. [6]

From the work done on [7] it is noted that the drone is forming its trajectory from the point cloud without hitting on the point cloud. This is done using Kd-Tree approach by taking the obstacle point clouds its neighbour points and as when the drone hits that particular point it deviates, forms a trajectory and reaches its destination.

III. Method

A. Detection

1) HSV: Fire detection can be done in many ways of which HSV is more accurate and easy to process based on the values/threshold of the color. HSV is an acronym for HUE, SATURATION, and VALUE. Each color will have its different threshold of which mild and dark yellow is segregated. Using two arrays of lower and upper threshold for yellow, we are performing HSV after converting the original image to RGB.

2) BGR $\rightarrow$ RGB : Converting the image to RGB will be the easiest task to perform HSV operation.

B. Tracking

1) Bounding Box: Using boundingRect, an inbuilt function of python is used to draw the bounding box around the fire detected. After drawn, a centre point is found for each image.

2) math.hypot(): Using math.hypot(), we found the distance between two centers i.e, two images of the fire detected. This process is processed for each image followed by the preceding image.

3) matplotlib(): When we plot all the points using matplotlib function of python, we would be able to visualize the tracking.

4) Unity 3D: For further observations, the data has been sent to a predefined host of unity 3D CORE using SOCKET library of python. We created an object fire, and developed a translation by different hierarchies. After which the center values is sent and observed the tracking in a detailed manner for further conclusions.

C. Data Collection

1) Point Cloud: We successfully created a point cloud of an entire apartment basement using a depth-scanning LIDAR sensor. We utilized the help of inbuilt LIDAR sensors in iPhones and using a third-party application Polycam, we were able to export the collected point cloud as **.ply** file

D. RANSAC

1) Detect Multiple plane: Now we are ready with sample data of a point cloud taken from a room basement with multiple exits. In order to compute points and work with it, we fitted the plane using the method called Random Sample Consensus(RANSAC). To get different planes, we used an api to detect multiple planes. There were 92 such plane equations when the function is implemented to fit the plane. We are in need of a plane which consists of exits. This is done by trial and error method on determining it from the 92 plane equations. Threshold value is given as lowest as possible, as because the more the lower the threshold value, the higher the best fit plane.

2) Pyransac3D: This is an inbuilt api to fit the plane using Random Sample Consensus. This method has an advanntage of structurized plane fitting for example if we have plane that is to be fit in the shape of cuboid, plane, cylinder or spherical this method can be done. Of implementing pyransac 3D with cuboid function, we would able to get the best fit of the exit planes but the major drawback is that as the exits are in the corner, when it is fitted against the cuboid shape it wasn't as best as it should be. Hence multiple plane detection is used for the approach of plane fitting. It was observed that each and every time when we run RANSAC api, it fits randomly and we get different output as when the kernel is restarted again and again.

E. Neighbor Points Detection

1) KD-Tree: One of the efficient approach to determine the nearest neighbor points in a point cloud is using Kd-tree algorithm. The nearest neighbour search algorithm aims to find the point in the tree that is nearest to a given input point. This search can be done efficiently by using the tree properties to quickly eliminate large portions of the 3D search space. We call the function to the point cloud data set used, and query it which in turn returns us the distance and indices of the nearest neighbor points.Now, with the indices we got, we find the corresponding point and the difference is calculated among them. If the difference is higher it is said to be exit for the best fit plane. There might be the case of the nearest neighbor to be calculated along the other axis. That particular point can also be assumed as the exit points. To make sure, we used searchknnvector3d() function of tree to trail and error the points. It means we use that function to find 100 nearest point with a point taken out randomly. By this way, we come to know the exit points and assigned to it two variables. As an example from our data point cloud, we have randomly taken 5000th point and coloured red(Fig. 1) of the nearby 100 points to make sure the assignment.

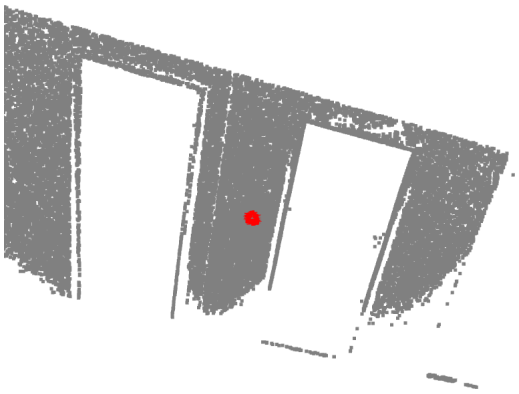

Fig. 1. Search knn vector 3D

2) Nearest-Neighbor: This method can also be done to compute nearest neighbor points from sklearn neighbors function module. This method return same as kd-tree approach. But if the next point and that corresponding point are said to have identical distances but different labels, the result will depend on the ordering of the training data.

F. Optimal exit determination

To determine the optimal exit, we calculate the difference between the fire (Centre point) with the two exit points simultaneously. Once the difference is measured, it is then compared to the exit points assigned as the variable. The algorithm works in such a way that if the difference between the centre of the fire point and any one of the exit is longer, that particular exit is said to be the optimal one. Once the optimal exit is known, the trajectory is formed to the exit determined.

IV. Experiments

- A video of fire spreading out is taken into picture for the experiment.
- After reading the video, random images are captured (4051 images in our case).In this example, a man lighting a paper using his lighter and it is spread out vertically. The image depicts the tracking of fire travelling via the down axis.
- When the video is read, it is stored in a folder and it is been masked out to detect the fire.
- After declaring two threshold values for the color we need, it is been ranged using inRange inbuilt python function for HSV operation.
- The HSV operation comes into play to detect the fire after converting all the images to RGB and performing bitwise AND between the HSV and the original image (Fig. 2).

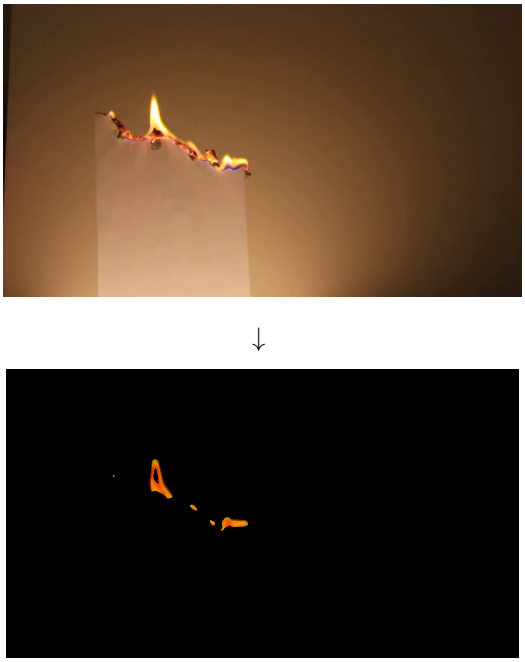

Fig. 2. Image to HSV

- After getting the final output, a rectangle/bounding box is drawn on the detected fire (Fig. 3) using boundingRect inbuilt function of python.
- Hence, we took number of videos, captured the image, and detected and tracked the fire which in turn achieved a good number of datasets
- Therefore, from the plot (Fig. 4), we could able to predict the direction of fire by tracking it.
- From the points obtained, distance is calculated from each centre point (Fig. 3) using math.hypot() function to analyze how it is been tracked.
- To be more specific, the usage of points on the bounding box will let us know on which direction the fire is spread in a fast manner.
- Likewise, We will design and train the model by providing different data sets and find the direction of fire.

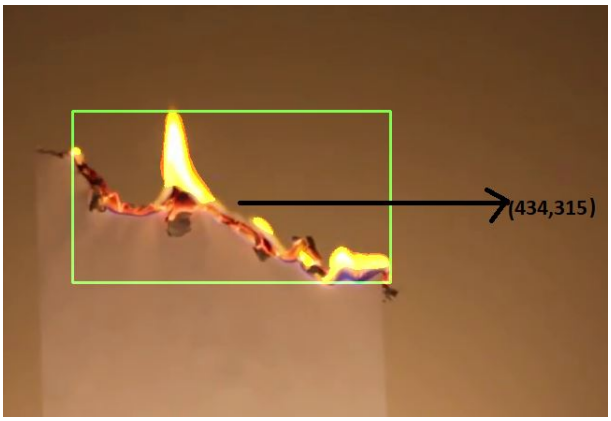

Fig. 3. Centre point of the bounding box

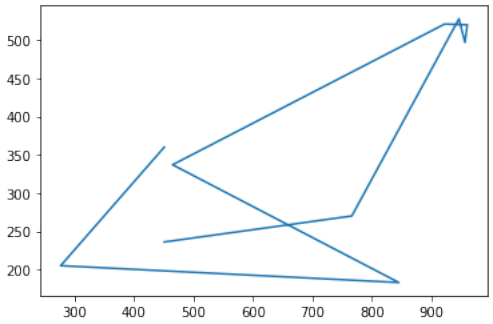

Fig. 4. Plot analysis of fire tracking

- The centre points are stored in an array and it is used as one of the datasets for machine learning.
- Using Linear regression, the model is trained for this dataset for prediction of the direction of fire and can be able to visualize how the data has been spread to learn more about it.Based on the direction of fire, the trajectory will be determined on the best exit path.
- A room map point cloud(Fig. 5) is taken using the iphone's LIDAR and converted to .ply format.

Fig. 5. 3D Room Map

- Multiple planes are detected using RANSAC(Random sample consensus) and it is expected to be 90 planes on our data set. A plane which consist of exits are taken for the approach of determining neighbour points.
- We are interested in the plane that consists of exit(Fig.

6) (the best fit plane).

Fig. 6. Best Fit Plane

- The neighbor points are determined using KD-Tree or any of the nearest neighbor algorithm. The distance of each points are calculated and the differences of each are found. A threshold is set for the difference to determine the exits.
- Once the exits are determined, the average mean is taken between the predicted position of fire with the exits. The lesser the difference, the fire is said to approach that particular direction and a trajectory is formed on the other exit(Fig. 7) .

Fig. 7. Optimal Exit

V. Conclusions

As mentioned previously, we were successfully able to train a created model which predicts the next fire spread trajectory coordinates from the trajectory dataset of detected fires using HSV and Bounding box method. Successfully able to implement RANSAC and KDTree to derive the exit points from the created point cloud of an apartment basement. Then using the created longest distance calculation algorithm, compared the distances of the exits from the fire coordinates to indicate the safest exit of all

The assumption and challenges we encountered during the project are that we did not have high sensitive LiDAR which can detect fire nor multiple calibrated cameras which can locate the fire in 3 Dimension. So we have detected and tracked fire using a 2D Camera and considered its Z-axis component to be a constant in point cloud. Next one coming to the creation of fire point cloud, the size and shape of the pointcloud keeps varying because of external factors as mentioned by Deligiannakis et al. [8]. Also since this is a closed room and is on a smaller scale, we considered fire spread to be as an ideal test case scenario omitting all other external factors.

For future work, we plan to run the fire spread trajectory coordinates prediction by taking into consideration of external factors as well such as ambient temperature, material on fire, moisture content in atmosphere, etc., We will check how good is our trajectory prediction in real world and how robust is our prediction system will be.

VI. Contribution

1) Prithvi Dev K V: Video dataset collection and programmed to detect HSV values of all the collected dataset videos frame by frame and calculated the coordinates of the detected fire and created the trajectories dataset. Did literature reviews of pointcloud algorithms and RANSAC. Captured pointcloud dataset of the basement using LIDAR and did RANSAC on the detected point cloud to retrieve multiple planes detected and find the optimal one out of all. Developed algorithm for optimal exit path calculation.

2) Thivish Naran: Video dataset collection and further made it visualize in Unity 3D. Trained the model for fire trajectory coordinate prediction. Did literature reviews of Point Cloud Visualiser and KDTree techniques for Nearest Neighbor Points detection. Implemented this technique across optimal planes on the pointcloud model and successfully able to retrieve the individual exit points from the pointcloud. Developed algorithm for optimal exit path calculation and trajectory based visualization on the pointcloud.

References

[1] P. Dai, Q. Zhang, G. Lin, M. M. Shafique, Y. Huo, R. Tu, and Y. Zhang, "Multi-scale video flame detection for early fire warning based on deep learning," *University of Alicante, Department of Computer Technology*, 2022. 1

[2] S. Wu and L. Zhang, "Posenet: Using popular object detection methods for real-time forest fire detection," in *Proc. Int'l Symp. on Computational Intelligence and Design (ISCID)*, 2018. 1

[3] A. Mouelhi, M. Bouchouicha, M. Sayadi, and E. Moreau, "Posenet: Fire tracking in video sequences using geometric active contours controlled by artificial neural network," 2020. 1

[4] A. Zarkasi, S. Nurmaini, D. Stiawan, Firdaus, Abdurahman, and C. D. Amanda, "Posenet: Implementation of fire image processing for land fire detection using color filtering method," in *Proc. IOP Conf. Series: Journal of Physics: Conf. Series 1196 (ICONISCSE)*, 2019. 1

[5] F. Khennou, J. Ghaoui, and M. A. Akhloufi, "Posenet: Forest fire spread prediction using deep learning," in *Proc. Geospatial Informatics XI*, 2021. 1

[6] M. S. Calvoa, J. A. Lopeza, A. F. Guilloa, and J. G. Rodrigueza, "Three-dimensional planar model estimation using multi-constraint knowledge based on k-means and ransac," 2017. 1

[7] Z. Zheng, T. R. Bewley, and F. Kuester, "Posenet: Point cloud-based target-oriented 3d path planning for uavs," in *Proc. Int'l Conference on Unmanned Aircraft Systems (ICUAS)*, 2020. 1

[8] G. Deligiannakis, A. Pallikarakis, I. Papanikolaou, S. Alexiou, and K. Reicherter, "Posenet: Detecting and monitoring early post-fire sliding phenomena using uav–sfm photogrammetry and t-lidar-derived point clouds," in *Proc. Advances in the Assessment of Fire Impacts on Hydrology*, 2021. 4