DroneVR: A Web Virtual Reality Simulator for Drone Operator

Vinh T. Nguyen
Dept. of Computer Science
Texas Tech University
Texas, United States
vinh.nguyen@ttu.edu

Kwanghee Jung
Dept. of Edu. Psychology & Leadership
Texas Tech University
Texas, United States
kwanghee.jung@ttu.edu

Tommy Dang
Dept. of Computer Science
Texas Tech University
Texas, United States
tommy.dang@ttu.edu

***Abstract*—In recent years, Unmanned Aerial Vehicle (UAV) has been used extensively in various applications from entertainment, virtual tourism to construction, mining, agriculture. Navigation, path planning, and image acquisition are the main tasks in administering these aerial devices in accordance with real-time object tracking for affordable aerial vehicles. Aircraft crash is one of the most critical issues due to the uncontrolled environment and signal loss that cause the aerial vehicle to hit the buildings on its returning mode. Furthermore, real-time image processing, such as object tracking, has not yet been exploited for a low-cost aerial vehicle. This paper proposes a prototype embedded in a Web-based application called *DroneVR* to mitigate the aforementioned issues. The virtual reality environment was reconstructed based on the real-world fly data (OpenStreetMap) in which path planning and navigation were carried out. Gaussian Mixture Model was used to extract foreground and detect a moving object, Kalman Filter method was then applied to predict and keep track of object's motion. Perceived ease of use was investigated with a small sample size users to improve the simulator.**

***Index Terms*—UAV, Virtual reality, Drone crash, 3D simulator, Openstreetmap, Path planning**

I. Introduction

According to the market research [12], Drone or (Unmanned Aerial Vehicle - UAV) sales have been increased significantly recently due to its lowering cost and mobility. As of 2018, there were more than 110,000 drones registered with Federal Aviation Administration (FAA) hovering on the sky, this number is more than double (2.5 times) compared to 2016 and is expected to exceed 600,000 by 2022. It is obvious that the use of drones has been utilized in many fields such as in construction, mining, agriculture, surveying, real estate.

Losing signal [10] or disconnected video transmission is one of the primary reasons causing drones to crash or lost. A modern light-weight drone is often equipped with a "Return Home" [18] function that allows the unmanned device to come back to its took-off point (considered as home) whenever its signal is lost. First, the drone flies up to a certain altitude - which is set by the operator; then it simply traces a straight line to its home without considering obstacle avoidance. This approach works well in an open area where no building or obstacle blocks the returning home way. On the other hand, the "Return Home" mode is more vulnerable to crash in a city. Drone operators need to clearly understand the environment - which is not always feasible as crashes did occur due to buildings hit [19]. Literature has made some efforts to alleviate this issue, for example, the DJI manufacturer (the most dominant manufacturers on the drone markets), provides a flight simulator [11] allowing drone operators to practice and get used to drone controller before the actual use. The limitation of this approach is that it does not reflect the real-world scenarios.

Object detection/tracking is another interesting topic which has been investigated in many fields, dedicated UAVs are equipped with sensors that are able to detect and track object, however, they are more expensive compared to regular ones. In the case of extreme conditions (e.g., crash, get lost, hijack), the accumulated cost could be high. There is a need to have cheap and affordable drones that are capable of doing same tasks and relies only on its camera only. Yet, one popular approach is to capture media stream from the UAV's camera then use machine learning method for object detection/tracking. However, this method requires objects to be trained before they can actually be used. TensorFlow [1] and OpenCV [5] are two great examples in this case and they have been used extensively for detect popular object such as people, bicycles, cars, trucks, animals, etc. Currently, TensorFlow is able to detect around 100 classes of objects via MobileNet feature extractor and OpenCV provides 80 classes through YOLO with a high prediction accuracy. However, when a desired object is not trained, it is a challenging task to leverage their strength. In addition, training object for model ready is another time consuming issue, transfer learning could be an alternative for using an existing pre-trained model but for a particular domain one may be interested in a few objects, thus taking into account of all classes in existing models is not preferable and computational expense.

This paper proposes an approach to address the aforementioned issue by constructing a virtual reality simulator that allows operators to get practice with drone in a mimic real-world environment before their devices actually being used. The scope of this work is to target cheap and affordable drone without modern sensors. The main contributions of this research thus are:

- It provides an approach to mitigate the risk of drone crashing with buildings by incorporating real-world map data into the VR system so that operators can practice

and get themselves familiarize with the environment;
- It gives a light-weight approach for object detection and tracking, taking into account of occlusion;
- It illustrates its approaches through an open-source virtual reality simulator called *DroneVR*; and
- It evaluates the *DroneVR* in terms of ease of use.

The rest of the paper is organized as follows: Sect. II summarizes similar work. We present our system design and simulation in Sect. III. A user study on the VR application is presented in Sect. IV. We conclude our paper with future work in Sect. V.

II. Related Work

In response to the need for controlling multiple UAVs with minimum personnel resources, Knutzon [17] created a virtual reality application that is capable of dynamic real-time path/re-path planning. In this early day, the shortage of available modern VR headset and controllers limits this application to be immersed only through stereoscopic binoculars, interaction with VR environment is thus restricted to 2D game-pad interface, the user was capable to selecting only pre-defined paths without the ability to manipulate them. The question of whether 3D immersive interface could provide any advantage over the traditional 2D interface still remains as indicated in their future work.

Crescenzio et al. [6] also designed a touch-screen based interface for UAV ground control station. High-level commands to the vehicle were sent from the touch screen and UAV operated in the 3D virtual environment. One interesting feature of this designed application is the overlay of 2D map in the background that makes the scene look realistic, incorporating 3D models is suggested to enhance the level of fidelity. This work concentrates on path planning for a generic vehicle model. The study result showed that 3D virtual environment conveyed the most important features such as the perception of the UAV's current state and the scenario at a glance. For targeting small drones (i.e., Bitcraze Crazyflie 2.0), Honig et al. [16] provided a testbed on how to link a small drone to a virtual environment instead of bigger and more expensive ones. Their work, however, mostly focused on testing algorithms rather than creating a user interface for drones. Paterson et al. [23] proposed an interesting work about setting up a path planning in a virtual reality environment by using Unity3D software in accordance with Oculus Rift headset. The idea was to translate or map user-defined paths from high-level (3dimension) to lower-level control (i.e., roll, pitch, yaw, and thrust). Their study result indicated a statistically significant improvement in safety and subjective usability over manual control. The above attempts are great examples of taking advantages of the virtual environment to understand, learn, and control the UAVs. However, **incorporating real-world data has not been fully explored or investigated**. This paper takes a further step by addressing the mentioned needs.

III. System Design

A. The DroneVR approach

DroneVR is developed using JavaScript libraries and in particular the ThreeJS [7]. Unlike other mentioned approaches that simulated VR environment in non-popular headset devices (i.e., Oculus Rift), our effort tried to bring VR experience to a more general audience with just a browser-supported device. Hence, our approach does not restrict users from depended on a single operating system. The use of web-based virtual reality has been studied extensively in [20], [22], [28]. In addition, the open-source platform would allow drone enthusiasts to make a similar experiment with any type of their drone. Real-time data (i.e., buildings locations and their heights) was retrieved from OpenStreetMap [14]. 3D models for our application were freely achieved from TurboSquid as suggested by Nguyen and Dang [21]. The primary goal of *DroneVR* is to create a VR application that presents drone operators a high-level view of the path planning/re-planning process. The *DroneVR* designing approach takes into account of the following considerations: (1) automated routing from a drone to a set of mission points, (2) semi-automated routing with the human in the loop 3) drone passively returning with object avoidance. To meet these goals, this paper proposes several features that are implemented in *DroneVR* based on the application design approach suggested by Shneiderman [25] for a visual application:

- **Overview Display (F1)**. Display overview of the 3D virtual environment.
- **Details on demand (F2)**. Present details of the planning paths construction with the perception of drone present in the scene.
- **Automated routing (F3)**. Automatically find the optimal paths among a set of mission points.
- **Semi-automated routing (F4)**: Re-routing optimal paths when operators gain control over the drone
- **Safety returning home (F5)**. Returning to the starting point (or pre-defined point) with object avoidance.
- **Object detection and tracking (F6)**: Tracking a moving object on a region of interest.

B. The DroneVR components

Based on the outlined features, *DroneVR* is designed with two main components: 1) The main component and 2) the navigation component.

1) The main component: This component represents the virtual reality environment in which 3d objects (buildings, drone, tank, and points of interest) are rendered (**Feature F1**). Since the proposed VR was implemented in the web environment, the computational expense should be taking into consideration. To improve performance, we used low poly 3D models with a minimum number of vertices and avoided using images as a texture, instead, we used normal color for filling the faces (i.e., drones, objects). For buildings, we do not render the faces (only vertices in wire-frame mode) to reduce occlusion and vision cluttering. OpenStreetMap

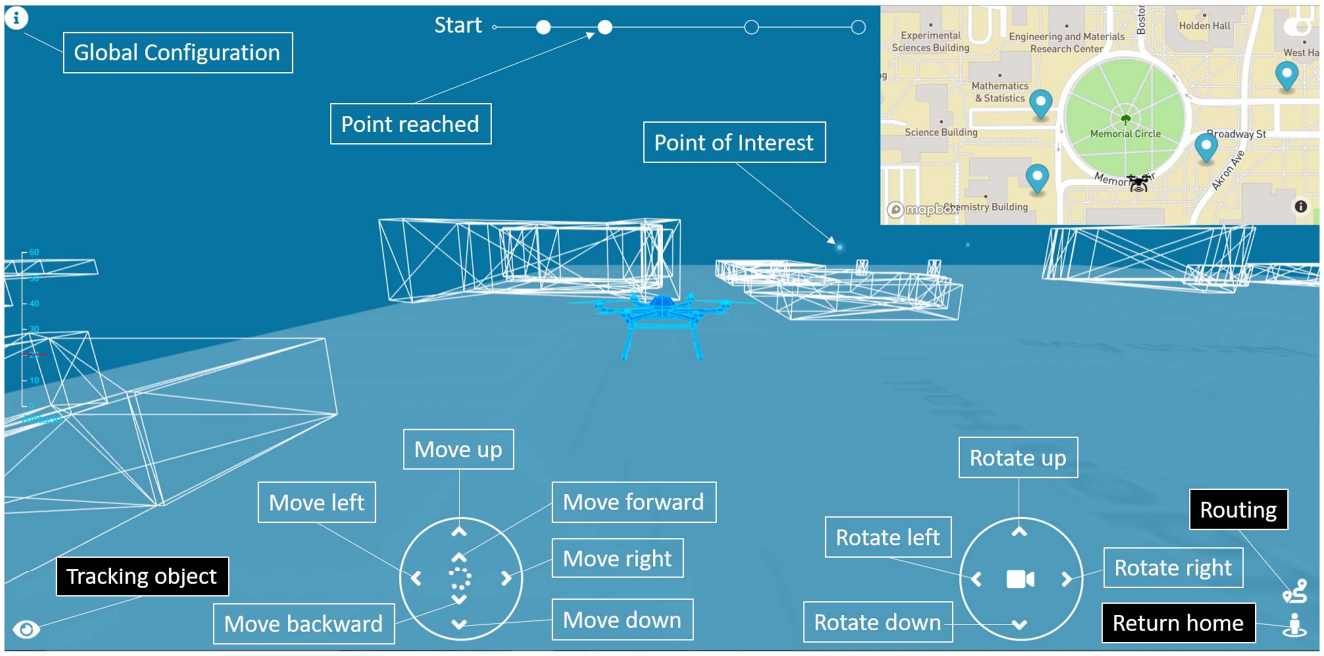


Fig. 1. *DroneVR* overview panel for a university campus map.

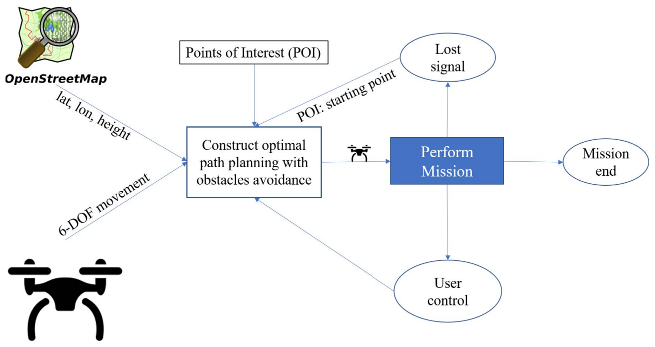


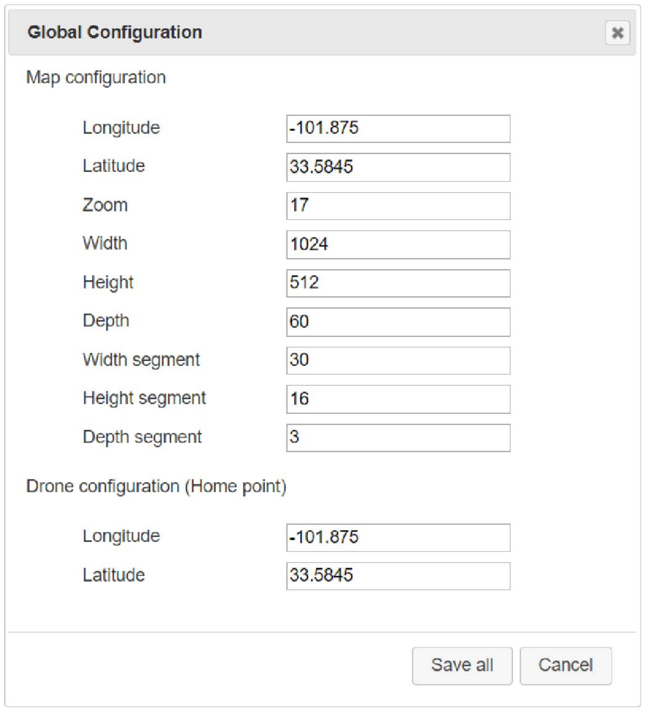
Fig. 2. The *DroneVR* architecture

provides data for each building in terms of points (or nodes) which hold information such as longitude, latitude, and height, this building data is mapped into a 3D virtual environment using Web Mercator Projection [3]. The formulas for the Web Mercator can be expressed as:

$$\begin{aligned} x &= \frac{256}{2\pi} 2^{zoomlevel} (\lambda + \pi) pixels \\ y &= \frac{256}{2\pi} 2^{zoomlevel} \left(\pi - ln\left[tan\left(\frac{\pi}{4} + \frac{\varphi}{2}\right)\right]\right) pixels \end{aligned} \tag{1}$$

where λ, φ are the longitude and latitude in radians respectively. Initially, users can configure the desired location of the 3D environment via Global Configuration panel (as depicted in Fig. 3). Center of the map was aligned with the center of 3d space.

2) The navigation component: This component contains five widgets: 1) a real-time thumbnail map, 2) two navigation controllers, 3) a mission points for routing, 4) a current drone altitude measurement and 5) a return home function.


Global Configuration

Map configuration

Longitude	-101.875
Latitude	33.5845
Zoom	17
Width	1024
Height	512
Depth	60
Width segment	30
Height segment	16
Depth segment	3

Drone configuration (Home point)

Longitude	-101.875
Latitude	33.5845

Fig. 3. Global configuration of the *DroneVR*

The real-time thumbnail map: is displayed at the top right corner of the *DroneVR* as depicted in Fig. 1 that gives users a high-level view of drone presence in the scene. It also enables users to set up the planning paths (**Feature F2**) or create points of interest by clicking on the map. Once, these POIs are generated, optimal path planning can be carried out by the "Routing" button shown in Fig. 5(c).

The path planning algorithm (**Feature F3**) was implemented involving two steps: First, it measures the shortest distance between the current drone position to all other POIs based on the notable A* (namely A star) search algorithm [15] which has been adapted in many current game engines for navigation (e.g., Unity3D). In this algorithm, we constructed a 3D grid system that spans the whole map, and the grid was divided into segments (width segment = 30, height segment = 16 and depth segment = 3 as shown in Fig. 3). The unreachable point on the grid is set if this point is inside/close to a building as depicted in Fig. 4 where the red dots are reachable and yellow dots are not reachable. These segment parameters are set due to the current map selection. If users prefer an area with more buildings, the resolution of the grid should be higher or vice versa.

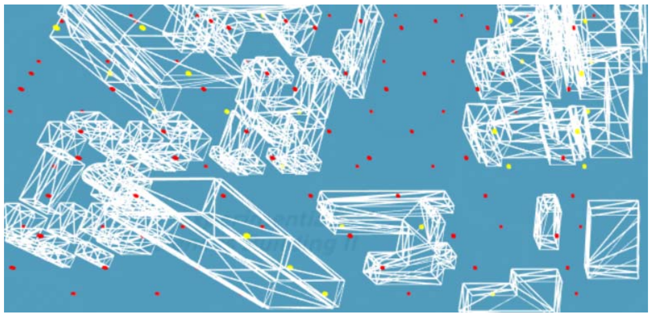


Fig. 4. Example 3D grid system of the *DroneVR*: red points are reachable whereas yellow points are unreachable

Based on the calculated distances, the second step involved optimal path planning with the traveling salesman problem (TSP) [4] algorithm. The ordered way-points was then visualized in the **the mission points indicator** widget located at the top middle screen. The main drawback of the thumbnail map is that it only gives the presence of a drone and POIs (longitude and latitude) in a 2-dimension space, and discards the third dimension (altitude). Instead, the **current drone altitude measurement** was created to keep track of the drone altitude at any given time as in Fig. 5(f).

There is a case when users want to manually control (**Feature F4**) the UAV to some points of interests from the start as usual operations or to navigate the drone to another point which is not set from the beginning. In this case, the **two navigation controllers** - as in Fig. 5 (d) and (e) - are provided that allow the operator to: move left, move right, move up, move down, move forward, move backward, rotate up, rotate down, turn left, and turn right. This option is triggered upon the manual mode (center of nav controllers) is pressed.

In case of losing signal or the operator want to take the drone back, the 'Return Home' function allows the UAV coming back to its starting point (**Feature F5**) which was set up in the Global Configuration panel as shown in Fig. 3. This option is represented in Fig. 5 (c). In this approach, the starting position will be used as a point of interest. The optimal path planing will be performed on a single point (from the current drone position to home point). By taking the knowledge of the environment, the chance of crashing will be reduced significantly.

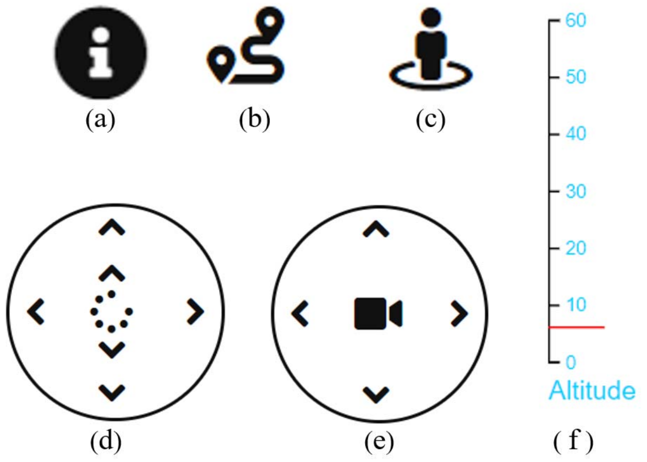


Fig. 5. The navigation control of the *DroneVR*: a) open global configuration, b) automated-path planning, c) 'Return Home' mode, d) movement controller, e) rotation controller, f) altitude measurement

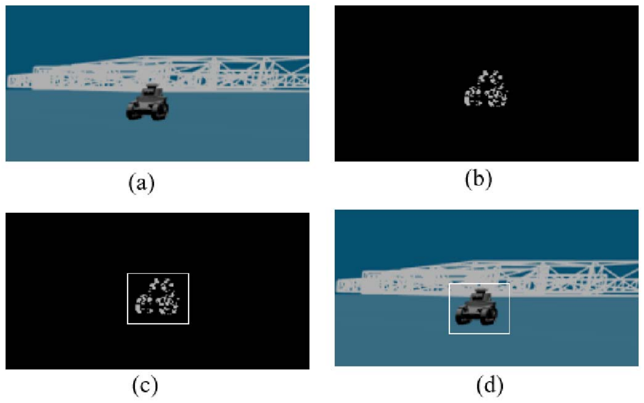


Fig. 6. Feature extractions and object detection: a) image captured in one frame, b) foreground extraction, c) blob analysis, d) detected object

3) Object tracking:

4) Foreground extraction: ThreeJS renders the 3D environment in a regular time interval (approximately 60 frames/images per second) for almost modern browsers. The image in each frame is color-encoded in RGB format (i.e., R-red, G-green, B-blue). To reduce computational time, the image was converted into a Gray-sale image with *luma* (brightness) color space which has been shown more relevant compared to other methods (i.e., hue, chroma) [13] for image compression and processing. The conversion is expressed as follow:

$$luma = 0.2989 * R + 0.5870 * B + 0.1140 * G \quad (2)$$

The value of luma color space ranges from 0 (black) to 1 (white). Once, the grayscale image was achieved, we used the Gaussian Mixture Model [27] to extract the foreground mask. The basic intuition behind using this model is that within a certain time interval, the brightness(luma value) of each image pixel does not change so much and often fall into its Gaussian distribution and this pixel value is considered as background pixel. The probability of the observing pixel at time t is described as:

$$P(X_t) = \sum_{n=1}^{K} \omega_{i,t} * \eta(X_t, \mu_{i,t}, \Sigma_{i,t}) \quad (3)$$

Where K is the number of Gaussian distributions (K = 5 in our VR application), $\omega_{i,t}$ is the weight of the pixel in the Gaussian i^{th} at time t, $\mu_{i,t}$ is the average value of the i^{th} Gaussian at time t, $\Sigma_{i,t}$ is the co-variance matrix of the i^{th} Gaussian at time t and η is the probability density function

$$\eta(X_t, \mu, \Sigma) = \frac{1}{(2\pi)^{\frac{n}{2}} \mid \Sigma \mid^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t-\mu_t)^T \Sigma^{-1}(X_t-\mu_t)} \quad (4)$$

At each time step t, the weight $\omega_{i,t}$, mean $\mu_{i,t}$ and the standard deviation for each Gaussian are updated accordingly. Since our 3D environment is not subjected to change (e.g, lighting, having debris which cause noise), the foreground feature can quickly be extracted with a low number of time steps (t = 5). In practice, one may consider to increase this number.

The foreground feature was represented by the foreground mask which contains logical values of 0 and 1 where 0 represents the background and 1 represents the foreground. Fig. 6(a) shows a single image captured at time t and Fig. 6(b) illustrates the foreground mask extracted at time $t = 5$.

5) Blob Analysis: The foreground mask extracted from the previous step was used for blob (binary large object) analysis. Our first approach was to use 4-connected component [9] method to detect a unique object - that is neighbouring pixels with the same color in four directions will form an object. However, this approach was not stable when we view the target object from different angles. Fig. 6(a) shows that the white pixels are clustered but not tightly connected. Filing holes [26] method thus can not be applied in this case due to no closed connected component. An alternative approach was employed based on the center of tendency in which a pixel belongs to an object if its distance to the object's center is less than a certain *threshold* (i.e., greater than 25). Fig. 6(c) depicted the extracted object (**Feature F6**) from a collections of disconnected pixels, the white rectangle is the bounding box constructed from the most *left, right, top, bottom* pixel positions. The bounding box is then applied back to the original image as shown in Fig. 6(d)

6) Kalman Filter: The detected object was tracked (**Feature F6**) in each frame/image based on the relative distance between the current object's position and its previous position, if the distance is relatively small or less than a *threshold*, the current and previous detected object are considered as the same object and is registered to a track. Here, we assume that the tank moves with a constant speed. In the blob analysis, our method to extract the blob was based on the center mean of tendency. The main drawback of this approach was that the center of the detected object may not be the same as the center of the real object because the detected object's mean was calculated based on the position of pixels. To overcome this issue, we applied the Kalman Filter algorithm [29] to estimate the correct position of the detected object. This algorithm uses detected, true and its predicted locations of the tracked objects as inputs then it estimates and justifies the predicted object position as close to its true value as possible. Another useful outcome that can be achieved from this algorithm was that future object position can be predicted with the absence of the true object in which the Kalman Filter considered the true object was the same as the predicted one.

IV. User Study

To evaluate the proposed VR application, we conducted a user study with 12 participants with respect to the ease of use - one component of the Technology Acceptance Model (TAM) [8]. TAM has proven to be a useful theoretical framework in explaining certain aspects of information technologies as well as understanding user behavior toward using these technologies. As a preliminary user study, we aimed to (1) investigate whether the user could learn and operate the simulated drone with ease, (2) access how easy for the operators to use our simulator and (3) obtain feedback from users experience to improve the usability of *DroneVR*. Before using the simulator, all participants were introduced the basic functionality of the tool either by direct communication or by watching the recorded video (made it available on YouTube [2]). Seven users come from the same departments with the authors, and five users are volunteers from the internet. Google form was used to collect the user's feedback. Time taken for doing the test is not limited since the application runs on the browser. Users can experiment and provide feedback at any time, but it was expected to get the feedback within a week.

Results: Overall, all users provided their feedback after doing their experiment. Besides the positive point of views, there were some issues with *DroneVR* such as: (1) the environment should be more realistic that includes building and road information instead of points and lines, (2) users should be able to modify the environment color, (3) the road is not clearly visible.

Discussion: As a preliminary user study to bring real-world data into virtual reality, there are some technical challenges in our proposed VR application. To increase the performance and smooth transition in the web-browser, we tried to minimize rendering cost and computations in each frame, so this was a trade-off between high fidelity and performance. In this work, users are only able to get a sense of the surrounding environment. To improve the fidelity, one approach is to use Mapbox GL to reconstruct the environment. However, the current version of Mapbox GL only supports pitching (tilting) of the map between 0 and 60 degrees from the plane of the screen. To be able to stand on the ground, around 90 degrees of pitching should be reached. Going in this direction, we were able to customize the Mapbox GL core and tilted the map to 90 degrees as depicted in Fig. 7. This new approach will be further examined in future work.

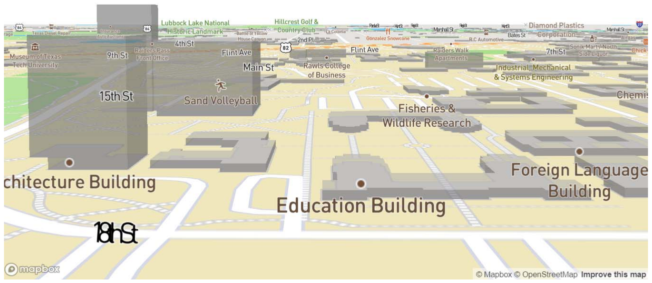

15th St
Sand Volleyball
Fisheries & Wildlife Research
chitecture Building
Education Building
Foreign Language Building
18th St
© Mapbox © OpenStreetMap Improve this map

Fig. 7. Pitching 85 degrees of the 3D environment in Mapbox GL

V. Conclusion and Future Work

This paper presented a drone simulator in a 3D environment that helps operators control and manipulate the drone before it is actually being used. Real-world map data was retrieved from OpenStreetMap. Path planning algorithm was adapted for optimal routing in 3D settings. Perceived on easy-of-use was evaluated to improve the simulator. This work-in-progress paper posed several limitations; (1) the sample size in user study is small, (2) the fidelity of the application is needed to be improved, and (3) no quantitative method was used to get insights of the user study. To overcome the limitations, a comprehensive user study will be conducted with adequate sample size using the extended technology acceptance model with task technology, perceived visual design, perceived usefulness, perceived ease of use, self-efficacy, and intention to use. Sensors on smartphone device can be utilized [24] to measure user's physical response after experiencing the application. Furthermore, the fidelity of *DroneVR* will be investigated by integrating the architecture into Mapbox GL in the future work.

References

[1] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*, pp. 265–283, 2016.

[2] Author. Dronevr: A web virtual reality simulator for drone operator, August 2019. https://youtu.be/ZOkB-_ck4lE.

[3] S. E. Battersby, M. P. Finn, E. L. Usery, and K. H. Yamamoto. Implications of web mercator and its use in online mapping. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 49(2):85–101, 2014.

[4] R. Bellman. Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM (JACM)*, 9(1):61–63, 1962.

[5] G. Bradski and A. Kaehler. *Learning OpenCV: Computer vision with the OpenCV library*. " O'Reilly Media, Inc.", 2008.

[6] F. D. Crescenzio, G. Miranda, F. Persiani, and T. Bombardi. A first implementation of an advanced 3d interface to control and supervise uav (uninhabited aerial vehicles) missions. *Presence: Teleoperators and Virtual Environments*, 18(3):171–184, 2009.

[7] B. Danchilla. Three.js framework. In *Beginning WebGL for HTML5*, pp. 173–203. Springer, 2012.

[8] F. D. Davis. *A technology acceptance model for empirically testing new end-user information systems: Theory and results*. PhD thesis, Massachusetts Institute of Technology, 1985.

[9] L. Di Stefano and A. Bulgarelli. A simple and efficient connected components labeling algorithm. In *Proceedings 10th International Conference on Image Analysis and Processing*, pp. 322–327. IEEE, 1999.

[10] DJI. 10 tips for preventing drone crashes, April 2018. https://store.dji.com/guides/drone-crash/.

[11] DJI. Dji flight simulator, May 2019. https://www.dji.com/simulator.

[12] DroneDeploy. 2018 commercial industry drone trends, May 2018.

[13] H. Fitriyah and R. C. Wihandika. An analysis of rgb, hue and grayscale under various illuminations. In *2018 International Conference on Sustainable Information Engineering and Technology (SIET)*, pp. 38–41. IEEE, 2019.

[14] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[15] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968.

[16] W. Hoenig, C. Milanes, L. Scaria, T. Phan, M. Bolas, and N. Ayanian. Mixed reality for robotics. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 5382–5387. IEEE, 2015.

[17] J. S. Knutzon. Managing multiple unmanned aerial vehicles from a 3d virtual environment. 2006.

[18] T. Levin. Return to home technology: Great idea or a drone killer?, November 2016.

[19] J. McDuffie. 15 unlucky drones that slammed into buildings, June 2017. https://www.popularmechanics.com/flight/drones/news/g3124/15-drone-crashes-buildings/.

[20] N. Nguyen, L. Virgen, and T. Dang. Hipervr: A virtual reality model for visualizing multidimensional health status of high performance computing systems. In *Proceedings of the Practice and Experience in Advanced Research Computing on Rise of the Machines (Learning)*, PEARC '19, pp. 130:1–130:4. ACM, New York, NY, USA, 2019. doi: 10.1145/3332186.3337958

[21] V. T. Nguyen and T. Dang. Setting up virtual reality and augmented reality learning environment in unity. In *2017 IEEE International Symposium on Mixed and Augmented Reality (ISMAR-Adjunct)*, pp. 315–320, Oct 2017. doi: 10.1109/ISMAR-Adjunct.2017.97

[22] V. T. Nguyen, R. Hite, and T. Dang. Learners technological acceptance of vr content development: A sequential 3-part use case study of diverse post-secondary students. *International Journal of Semantic Computing*, 13(03):343–366, 2019. doi: 10.1142/S1793351X19400154

[23] J. Paterson, J. Han, T. Cheng, P. Laker, D. McPherson, J. Menke, and A. Yang. Improving usability, efficiency, and safety of uav path planning through a virtual reality interface. *arXiv preprint arXiv:1904.08593*, 2019.

[24] S. E. Seo, F. Tabei, S. J. Park, B. Askarian, K. H. Kim, G. Moallem, J. W. Chong, and O. S. Kwon. Smartphone with optical, physical, and electrochemical nanobiosensors. *Journal of Industrial and Engineering Chemistry*, 2019.

[25] B. Shneiderman. The eyes have it: A task by data type taxonomy for information visualizations. In *The craft of information visualization*, pp. 364–371. Elsevier, 2003.

[26] P. Soille. *Morphological image analysis: principles and applications*. Springer Science & Business Media, 2013.

[27] C. Stauffer and W. E. L. Grimson. Adaptive background mixture models for real-time tracking. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, vol. 2, pp. 246–252. IEEE, 1999.

[28] V. T. Nguyen, R. Hite, and T. Dang. Web-based virtual reality development in classroom: From learner's perspectives. In *2018 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*, pp. 11–18, Dec 2018. doi: 10.1109/AIVR.2018.00010

[29] G. Welch, G. Bishop, et al. An introduction to the kalman filter. 1995.