AR for UAV-Based Image Acquisition

Henry Ang, Jingran Zhou

Introduction

Computer Vision models have been gaining significant visibility in the industry in the past decade. The success of these modern computer vision applications largely stems from the usage of deep neural networks. It is a consensus that training deep learning models requires a tremendous amount of training data. In addition, manual labeling of the training images is necessary for supervised learning. One salient disadvantage of such traditional "crowdsourcing" methods of image collection and labeling is the lack of *personal* training examples. For example, a well-trained object recognition model can easily identify images of dogs, but it probably cannot find the image of a specific dog among all the dog images. In other words, most existing datasets contain training examples of *generic* categories, which may not be suitable for individual-level objects. As a solution, this project provides an interface that allows users to easily collect training images, so that data of more fine-grained categories can be readily produced.

We base our project on an existing approach called LabelAR, an interface that enables efficient collection and labeling of training images for object detection algorithms. LabelAR allows its user to place 3D bounding volumes upon the objects of interest using an augmented reality compatible mobile device. Based on the bounding volumes, 2D bounding boxes are generated on the images, achieving automatic labeling. However, LabelAR is only applicable to small-scale objects placed on a flat surface due to its dependency on a mobile device. Physical limitations of human movements prohibit the application of LabelAR on large objects (e.g., buildings, trucks). To break through such limitations, we propose using **drones** to collect images of large objects.

Our project has two areas of focus -- Human-Computer Interaction (HCI) and Computer Vision (CV). Regarding HCI, we have developed a user interface that can 1) easily control the drone and 2) quickly collect images from various viewpoints; In terms of CV, we developed a bounding box generator capable of 1) estimating the position and size of target objects and 2) automatically generate bounding boxes around target objects. Due to the time constraint and the fact that we are a two-person team, our project is completed in Unity simulation. Our deliverables include a UI for drone control and image acquisition, alongside a bounding box generator. We have also evaluated the efficiency of our solution based on three timed experiments to be discussed below.

Related Work

There are three areas to focus on when reviewing prior work: image collection, user interface for 3D content and robotics for data collection.

We first review the works related to image collection. The target of most prior projects related to image collection is to reduce the collection time of training images, including the time to capture images and the time to label the images. Some research explores methods to shorten labeling time by combining the two steps. Yeh [8] proposes to take two images, one with and the other without the target object, to automatically calculate the label for the object, so that labeling does not require human effort. In LabelAR, Smith [5] suggests an AR interface based on phone cameras for collecting training images. The mobile application presented in the paper allows the user to manually adjust the size of the bounding box of the target object and then uses 3D markers to guide the user to take pictures of the target object from different angles. After the image collection process, human effort to label bounding boxes is not needed because 2D bounding boxes are automatically generated from 3D bounding boxes. In our project, the key difference is the movement of the camera. While in LabelAR, the camera is on the phone and moves with the user, in our project, the camera is on a UAV, and the UAV can either be controlled remotely or move automatically along a planned route.

Then, we review the works related to 3D user interface. Some early works use special devices to connect the user with the virtual 3D space. Tsang [6] created a flat-panel display connected to a mechanical arm that has six degrees of freedom and used it to display the virtual 3D scene on a one-to-one scale. More closely related to our project, Funk [1] discusses the various user interfaces for controlling drones, such as the conventional remote controller with joysticks and the more recent gesture control by hand tracking. In our project, the user interacts with the drone by an iPad using the touch screen. The user interacts with the virtual joysticks like a physical drone controller. Another aspect of the user interface related to our project is 3D object creation. Lau [2] proposes using primitive 3D shapes for users to create 3D objects in the virtual space. In our project, we use rectangular 3D boxes as bounding boxes because they can be easily translated to 2D bounding boxes. Our contribution is that we allow users to draw arbitrary shapes and use computer vision algorithms to produce 3D bounding boxes automatically.

Lastly, we review the works related to robotics for data collection. Prior works in this area are often related to visual recognition. Welke [7] proposed the automatic acquisition of an object's visual representation by rotating the held object. Other projects [3,9] use convolutional neural networks to identify the pose of the target object and then navigate the robot to perform related tasks. In the realm of drones, Roberts [4] proposed an algorithm to automatically generate the drone trajectory for collecting information-rich images in order to produce accurate reconstructions of large-scale 3D scenes. In our project, we will focus on collecting images from multiple angles and overlaying accurate bounding boxes on objects. We aim to collect images that are suitable for machine learning algorithms.

Implementation

Figure 1 is our system diagram. The user, through the UI, interacts with the simulation. The simulated drone outputs its spatial location and a depth map produced by a depth camera onboard. With this output information, the bounding box generator can render rectangles on certain objects. These rectangles are the labels for identifying the corresponding objects. We primarily focus on two components: the **UI** and the **Bounding Box Generator**. Additionally, we also implemented a simple autopilot feature in the drone movement

control module to automatically collect images from a variety of viewpoints by rotating around the target objects.

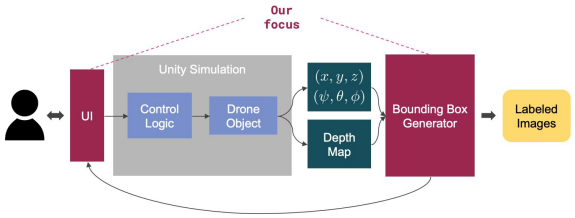

Figure 1: System Diagram

Although we base our project in simulation, we argue that our work is also applicable to real drones. Firstly, our code for the UI and the Bounding Box Generator has clearly defined API's with generic parameters, so these two components are highly modular and can be plugged into other systems easily. Secondly, while it is true that our drone control logic is written in C#, it is not dependent on any C#-exclusive packages. Thus, the control logic can be translated into other languages with minimal effort.

Next, we discuss how our UI allows the user to select objects by drawing arbitrary shapes, followed by a thorough examination of the core component of this project -- the Bounding Box Generator.

User Interface

Our UI consists of three parts: **drone control**, **bounding boxes**, and **object selection**. Figure 2 shows all three components.

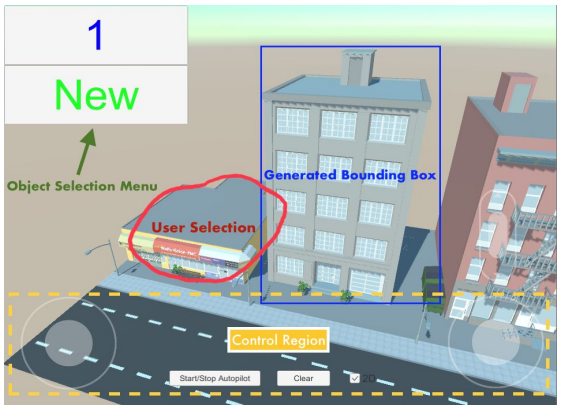

Figure 2: User Interface

First, our drone control UI (marked in yellow) is similar to real drone controllers. The left and right joystick control horizontal translation and rotation, respectively. Note that only yaw and pitch are supported (without roll) because a drone needs to maintain levelness during its flight. The slide-bar above the right joystick changes the elevation of the drone. Furthermore, we have introduced minute random perturbations to the drone to make the simulation more realistic. Besides, there are three options at the bottom: the leftmost button toggles our autopilot feature; the "Clear" button erases all the bounding boxes; the "2D" toggle determines if the bounding boxes are displayed as 2D rectangles or 3D volumes.

Second, the blue rectangle is a bounding box, whose generation is discussed in the next section. The 2D box in the screenshot is based on a 3D volume, which tightly encases the building, approximating the outline of the building. This volume can be seen with the "2D" toggle off. The 2D rectangle is obtained by projecting the vertices of the 3D cube from the world space onto the screen space. By calculating the extrema of the points' x and y values on the screen, we can construct the 2D box in the GUI.

Third, the user can "circle" objects to select them. The user can draw *arbitrary* shapes on the screen to indicate the objects he/she wants to label (shown in red). After drawing the shape, a pop-up menu will appear on the top-left corner, asking the user if this shape corresponds to existing objects or a new object. This correspondence is color-coded. For example, the *blue* "1" in the screenshot corresponds to the building inside the *blue* bounding box. Selecting this option will associate the red user drawing with the blue building, which is undesirable. However, if the user clicks on the "New" button, the system will create a new object, color-coded in green, resulting in two bounding boxes for two buildings.

It is worth noting that when the user draws shapes on the screen to select objects, these shapes will not necessarily be valid polygons. Therefore, it is our responsibility to approximate a reasonable boundary of the user's selection. The classic convex hull algorithm is not suitable for this application because it prevents the user from circumventing certain regions. Consequently, we chose *concave* hull, an algorithm for approximating concave polygons. With our fine-tuned concavity and scale factor parameters, the algorithm is surprisingly robust, capable of generating reasonable hulls regardless of the validity of user drawings.

Bounding Box Generator

We assume that our drone is equipped with a depth camera. By cropping the depth map based on the region selected by the user, we obtain a point cloud. We can generate a 3D bounding box from the point cloud simply by calculating its extrema on the x, y, and z-axis. However, the user's drawings are likely imperfect, and naturally the point cloud is noisy. Therefore, to get an accurate bounding box, we need to filter out points in the point cloud that do not belong to the object selected by the user.

Initially, we used a simple Gaussian filter shown in Figure 3. It filters out points too far away from the centroid of the point cloud. Each of the x, y, and z-axis has its own filtering threshold. For example, the threshold for x-axis is $\sigma_x \cdot t$, where σ_x is the standard deviation of the x-values of the points, and t is a predefined constant. However, this approach performs poorly, as users often have to select the same object multiple times for a complete bounding box.

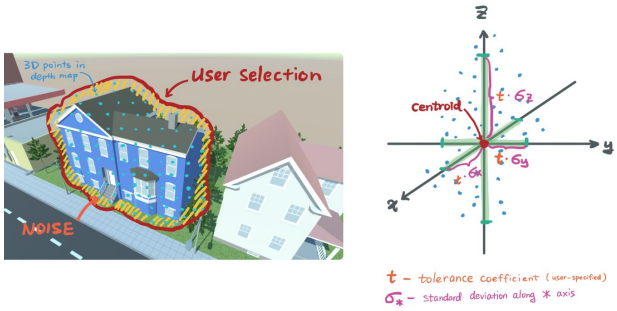

Figure 3: Gaussian Filter

Fortunately, we discovered another filtering algorithm that significantly outperforms the above method. Based on the feedback we received in our presentation, we designed a connectivity-based filtering algorithm, which is visualized in Figure 4.

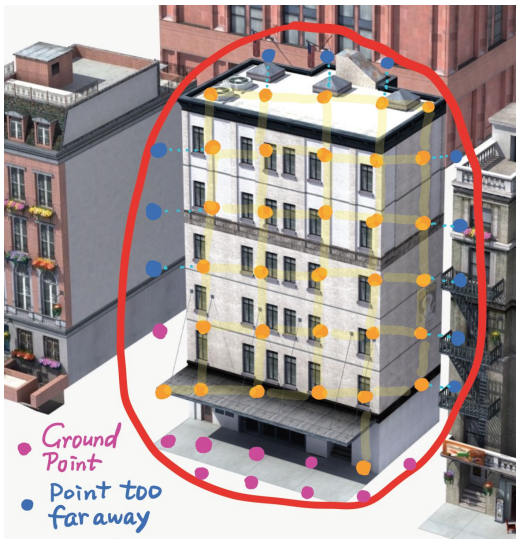

Figure 4: Connectivity-Based Filtering

Firstly, we perform ground detection to remove all the ground-level points (in purple). This disconnects the buildings from each other. Secondly, with the remaining points, we initiate a Depth-First Search (DFS) from the point that is closest to the centroid. This is because we hypothesize that the centroid is likely within our target object. In the DFS, two points have to be spatially close enough to be considered "connected." For instance, the blue points in the figure are too distant from the orange ones to be "connected" because they belong to another building. The minimum distance between two points for them to be "connected" is defined as the

connectivity threshold, which is adjustable. Finally, the connected component resulting from the DFS (in orange) constitutes our desired point cloud, with which we can generate a bounding box.

Evaluation

In order to evaluate the performance of our project and whether it meets our expectations, we carried out experiments to measure quantitative and qualitative results. We asked 8 participants (Students of UC Berkeley) to perform object labeling tasks in our simulation environment under different experimental settings and measure the completion time of different tasks to evaluate performance.

In our study, the independent variables are: 1) The number of target objects to label, 2) The input device and 3) The use of automatic drone navigation. In each experiment, we ask the participants to perform the task twice, labeling 1 object and 3 objects respectively (Figure 4). Table 1 illustrates the experiment settings of the three experiments.

Figure 4: One Building and Three Buildings to Label

Experiment No.	1	2	3
Input Device	Keyboard & Mouse	Touch Screen	Touch Screen
Automatic Navigation Enabled	No	No	Yes

Table 1: Experiment Settings

Before each experiment, participants were given sufficient time to familiarize themselves with the system. Then, the time taken to perform each of the labeling tasks was recorded. The criterion for task completion is that each 3D bounding box covers 90% of the target building's volume.

Quantitative results. Figure 6 and 7 illustrates the task completion times and average task completion time under different settings. The quantitative results imply that the touch control interface outperforms the keyboard-and-mouse interface, and auto-navigation is faster than manual navigation.

In the first two experiments, user attention is required throughout the collection process. In the third experiment, however, the user will not need active attention when the drone is on autopilot. We found out that on average, human attention is required 47% of the time for the one-building task and 57% of the time for the three-building task.

Figure 6: Time to Label Three Buildings Against Time to Label One Building

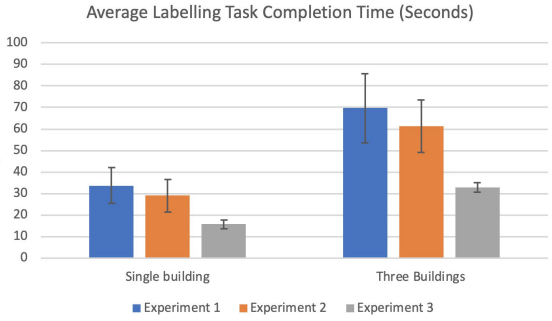

Figure 7: Average Task Completion time under three experiment settings

Qualitative results. From observing the participants performing the different tasks, we found the biggest component of completion time to be drone navigation. The time it took for the participants to navigate the drone to the desired position is much longer than that of drawing outlines of buildings or interacting with UI elements.

The participants reported that virtual joysticks on the touch screen are more intuitive than keyboard control. Because the virtual joysticks closely resemble the controls in video games, participants found it more natural to control the drone with touch control.

Discussion

Bounding Box Accuracy. Through the development of this project, we have improved the algorithm for bounding box generation a few times. During our experiments, we found the generation of the bounding boxes to be generally fast and accurate, given the available information from the depth camera. On average, it takes less than manual two selections for the automatically generated bounding box to cover more than 90% of the volume of the target object. One limitation we have identified in the latest iteration of our bounding box generation algorithm is that it ignores small sections of a build when the connection between them and the larger section of the building is hidden. For instance, in our sample scene, the gas station has a large rectangular roof and relatively small pillars and gas pumps. During annotation, the structures that connect pillars to the roof are covered by the roof and thus not available to our algorithm. Because our algorithm is based on the connectivity between points in the point cloud, the pillars are considered disconnected, and the generated bounding box only contains the roof. The user has to make separate selections of those smaller parts in order to make the bounding box cover the entirety of the building.

Collection Time. As we improve our user interface, we found the collection time to substantially decrease. The major bottleneck of collection time is drone navigation. With the improved algorithm, we reduce the number of manual object selections it takes to generate satisfactory 3D bounding boxes. Thus, users need to navigate the drone to fewer positions and the collection time is also reduced. Additionally, because of auto-navigation, drones can self-locate to different positions faster. However, the current auto-navigation has a relatively naive implementation. The drone rotates around the target objects at a constant height, so the positions the drone reach are not necessarily the best for object selection. Another potential drawback of the current routing strategy is that when multiple target objects are present, the drone chooses the center of all the objects as the center of rotation, and this route is likely to be suboptimal for individual objects. When target objects are close to each other, large portions of back objects will be blocked by front objects. Either manual navigation or a more advanced route planning algorithm is needed to solve this problem.

Conclusion

To fine-tune pretrained computer vision models for predicting custom image classes, we need additional *labeled* training images. However, existing image acquisition approaches are often limited to hand-held devices with little automation. Thus, we propose a new method for collecting and labeling images automatically using a drone with a depth camera. Our major deliverables include a user interface that enables the user to select

multiple objects by drawing arbitrary shapes and a mechanism to automatically generate bounding boxes around the selected objects. We also developed an autopilot feature to rotate the drone around the objects to capture images from a variety of angles. To evaluate our work, we conducted three timed experiments. We discovered that our interface has the highest image collection efficiency on a touchscreen with autopilot enabled. Furthermore, our analysis suggests drone navigation to be the main bottleneck. Therefore, optimal UAV path planning could be a promising venue for future endeavors.

Bibliography

1. M. Funk. "Human-Drone Interaction: Let's Get Ready for Flying User Interfaces!". In: *Interactions*, vol. 25, no. 3, 2018, pp. 78–81., doi:10.1145/3194317.
2. M. Lau, M. Hirose, A. Ohgawara, J. Mitani, and T. Igarashi. "Situated Modeling: A Shape-stamping Interface with Tangible Primitives". In: *Proceedings of the Sixth International Conference on Tangible,Embedded and Embodied Interaction*. TEI '12. ACM, Kingston, Ontario, Canada, 2012, pp. 275–282. doi: 10.1145/2148131.2148190.
3. J. Leitner, A. Förster, and J. Schmidhuber. "Improving robot vision models for object detection through interaction". In: *2014 International Joint Conference on Neural Networks (IJCNN)*, 2014, pp. 3355–3362.
4. M. Roberts, et al. "Submodular Trajectory Optimization for Aerial 3D Scanning." In: *2017 IEEE International Conference on Computer Vision*, 2017
5. J. Smith, M. Laielli, G. Biamby, T. Darrell & B. Hartmann. " LabelAR: A Spatial Guidance Interface for Fast Computer Vision Image Collection". In: *Technical Report*, No. UCB/EECS-2019-58, 2019
6. M. Tsang, G. W. Fitzmaurice, G. Kurtenbach, A. Khan, and B. Buxton. "Boom Chameleon: Simultaneous Capture of 3D Viewpoint, Voice and Gesture Annotations on a Spatially-aware Display". In: *Proceedings of the 15th Annual ACM Symposium on User Interface Software and Technology*, 2002, pp. 111–120. doi:10.1145/571985.572001.
7. K. Welke, J. Issac, D. Schiebener, T. Asfour, and R. Dillmann. "Autonomous acquisition of visual multiview object representations for object recognition on a humanoid robot". In: *2010 IEEE International Conference on Robotics and Automation*, 2010, pp. 2012–2019.
8. T. Yeh and T. Darrell. "Doubleshot: An Interactive User-aided Segmentation Tool". In: *Proceedings of the 10th International Conference on Intelligent User Interfaces*. IUI '05. ACM, San Diego, California, USA, 2005, pp. 287–289. doi: 10.1145/1040830.1040901. url: http://doi.acm.org/10.1145/1040830.1040901.
9. A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao. "Multi-view self-supervised deep learning for 6D pose estimation in the Amazon Picking Challenge". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1386–1383.