SSS: Towards Autonomous Drone Delivery to Your Door Over House-Aware Semantics

Shengqing Xia
Department of Computer Science,
Purdue University
West Lafayette, IN, USA
xia170@purdue.edu

Junpeng Guo
Department of Computer Science,
Purdue University
West Lafayette, IN, USA
guo567@purdue.edu

Chunyi Peng
Department of Computer Science,
Purdue University
West Lafayette, IN, USA
chunyi@purdue.edu

ABSTRACT

In this work, we present our attempt to tackle the *last-hundred-feet* problem for autonomous drone delivery. We take a computer-vision-based approach to progressively landing towards a convenient and safe drop-off point at all times (here, at the front/garage door). Specifically, we develop structural semantic segmentation (SSS), a new technique that leverages a single-family house structure to streamline and enhance semantic segmentation in the drop-to-door problem context. We implement SSS into an Android app; Our preliminary evaluation in a residential zone shows SSS is promising to make autonomous drop-to-door in real-time, with no need to wait for slow visual processing.

Video demo is available at Youtube [5]. App is released at Github [6].

CCS CONCEPTS

• **Computer systems organization** → *Robotic autonomy*; • **Computing methodologies** → *Computer vision*; **Vision for robotics**; **Image segmentation**.

KEYWORDS

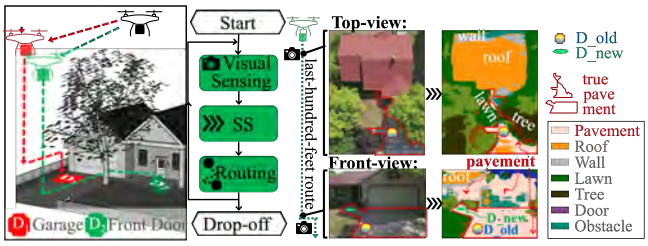
Drone delivery, Semantic segmentation, Structural semantic segmentation (SSS)

ACM Reference Format:

Shengqing Xia, Junpeng Guo, and Chunyi Peng. 2024. SSS: Towards Autonomous Drone Delivery to Your Door Over House-Aware Semantics. In *The 25th International Workshop on Mobile Computing Systems and Applications (HOTMOBILE '24), February 28–29, 2024, San Diego, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3638550.3641129

1 INTRODUCTION

Drone delivery is gaining momentum. By delivering packages through unmanned aerial vehicles (aka, drones), it is potentially disrupting last-mile delivery with enormous convenience and advantages (e.g., speedy delivery, reduced costs, easier access to rural areas, and good for environment) [24, 25]. Recent years have witnessed intensive R&D efforts from industry and research community, including 660,000 commercial drone deliveries to customers and countless test flights in the field [10]. The global drone delivery market expects to grow by half every year, hitting $9.2 billion in 2027 from $2.1 billion in 2023 [25].



(a) Illustration (b) Flow (c) A real-world instance over SS

Figure 1: An illustration of the last-hundred-feet delivery-to-door solution over semantic segmentation.

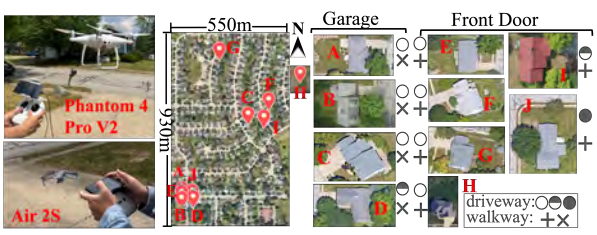
Drone delivery is not easy, particularly dropping off the package at the door of a residence (like FedEx couriers) in a fully autonomous manner. Most prior efforts are centered on *the last-mile problem*, where the drone flies from the source to the destination house through advanced route planning and navigation [7–9, 22, 23, 27]. Different from them, we target at *the last-hundred-feet problem*, which aims to land at a specific-and-small point (marked as "**D**", say, in front of the garage/front door) of a residential house, instead of any point around the destination house. As illustrated in Figure 1, the drone starts with any point above the destination house (say, 100+ feet above) and needs to progressively locate and fly towards a drop-off point (D-point) convenient for customers until it finally lands at the last D-point *safely* (no damage to the drone and surrounding environment), *precisely* (at a foot-level or meter-level), and *quickly* (say, within tens of seconds).

A straightforward solution is to leverage semantic segmentation (SS), a popular computer vision technique [21, 28], to recognize meaningful segments (say, roof, lawn, wall, door, pavement, and other house features) out of a collection of pixels (say, an image or a video frame captured by the drone) and determine or update the D-point. As the drone flies towards the current D-point, the view captured by the drone changes and thus the D-point might be updated accordingly. For example (Figure 1c), the drone later sees a better D-point at the front door, which is invisible from the initial top-view but becomes visible till the drone lowers below the tree. The SS-based solution runs recursively in many rounds. In each round, the drone performs visual sensing, SS and routing (flight control) and repeats this process until it finally lands at the last D-point. In this work, we target at the D-point in front of the garage/front door of a single-family house (SFH), which is the dominant housing type in the USA [4].

However, such SS-based solution faces two practical challenges (§2). *First,* SS *is not accurate enough to determine the* D*-point while*

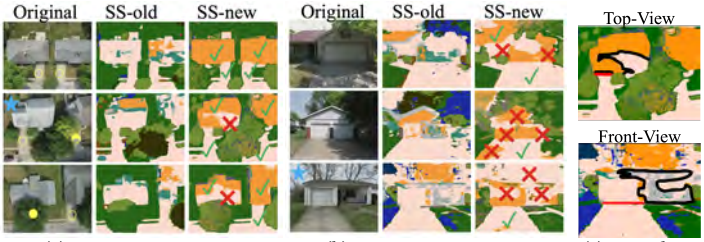
HOTMOBILE '24, February 28–29, 2024, San Diego, CA, USA

ACM ISBN 979-8-4007-0497-0/24/02.
https://doi.org/10.1145/3638550.3641129



(a) Test kits (b) Map (c) 10 SFHs

Figure 2: Experimental settings.



(a) Top-view instances (b) Front-view instances (c) Boundary

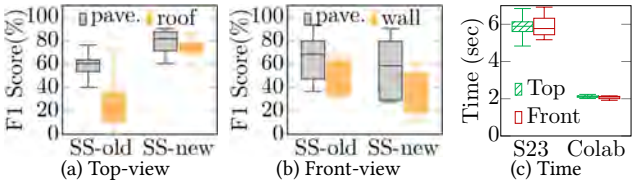
Figure 3: The instances with SS results and extracted boundaries.

meeting the precision requirement. As illustrated in Figure 1c, SS fails to get the right segment boundary (here, the door/wall in the front-view recognized as the pavement); It is impacted by environmental factors such as sunshine, shadow, weather and color space, etc. *Second,* SS *is compute-intensive and very slow.* It takes several seconds to run SS once, particularly on resource-constrained devices like smartphones or drones. This makes real-time visual sensing and flight control hard, if not impossible. As a result, the drone highly likely needs to hover until SS completes (for the sake of safety). We notice that these practical issues have not been disclosed or addressed in related work [11, 14, 18–20], which use SS to determine the landing point; They all are based on theoretic models or simulations, without taking practical factors into account. More details are given in our preliminary study (§2).

In this work, we attempt to tackle practical challenges to develop a SS-based solution for the last-hundred-feet problem. By this means, the developed solution can work with mid-end or even low-end drones without requiring hardware upgrades. Our design is driven by the following heuristics. *First, there is no need to run the legacy* SS *to accurately recognize all the classes supported.* It is because the D-point is mostly on the pavement (driveway/walkway in front of the garage/front door). We should focus on semantic segments of our interest. Precisely, we should locate and track the adjacent boundary between the house and its pavement (driveway/walkway), namely, the one between the roof and the pavement in the top-view or the one between the door/wall and the pavement in the front-view. *Second, there is no need to run* SS *every time to determine/update the* D*-point.* A house structure does not change although the view varies when the drone drops from the above. Stationary objects in the physical world can be exploited to streamline visual processing across multiple views over time. More essentially, a SFH follows a common structure (e.g., the driveway starts at the road and ends at the garage, the walkway ends at the front door); Inspired by these, we develop *structural semantic segmentation* (SSS) over prior-known SFH structure customized for autonomous delivery to the door (§3). We make use of the house structure and build SSS on top of the legacy SS solution with three main components: (1) a quick patch over SS to handle a single view, (2) a lightweight update and tracking across multiple views and (3) a fast algorithm to detect movements (say, moving cars or pedestrians). More details are elaborated in §3.

We have implemented SSS and integrated it to an Android app to run delivery-to-door experiments in the field (§4). With SSS, the drone successfully lands at the front/garage door of all the participating SFHs in a residential zone in West Lafayette, IN.

Release: Demo videos are at [5] and Android app is at Github [6].



(a) Top-view (b) Front-view (c) Time

Figure 4: Comparisons of SS-old and SS-new in terms of accuracy (F1 score), as well as the consumed time.

2 MOTIVATION: FROM SS TO SSS

We use a preliminary study over a residential zone to present the limitations of the conventional SS solution and motivate our design.

Methodology. We use two drone models of DJI Air 2S and DJI Phantom 4 Pro V2 to collect the top/front views of target SFHs in a residential zone of about 240 SFHs in West Lafayette, IN (Figure 2b). We capture the top views at an altitude of 100–150 ft above all the SFHs and the front views at a distance of 3–10 ft away from the garage door of 10 participating SFHs (with the consent from house owners, Figure 2c). These SFHs have representative house plans with various driveway visibility from the above: ○: fully visible, ◐: partially visible, ●: almost invisible (more details in §4). We use the SS model developed for aerial drone images [2], which supports 24 classes and is trained by a public dataset [3]. We find that this SS model (denoted as "SS-*old*") performs poorly. We then add 50% of images randomly selected from our dataset to train the SS model (denoted as "SS-*new*") and use the rest images for testing. By default, we run SS over an Android smartphone, Samsung Galaxy S23 Ultra with Adreno 740 (GPU) and Octa-core (CPU). To compare time consumption, we also use Google Colab with A100 (GPU) [15].

Results. Figure 3 shows several illustrative instances and Figure 4 shows the results in terms of accuracy and overhead. We have three observations. *First, the conventional SS performs poorly but can be significantly improved through proper training.* SS-new outperforms SS-old in the top-view cases (Figure 3a and Figure 4a); For instance, the F1 score for one class of "roof" roars from 20% to 74% (median). It matches with our common expectation: new training samples can be effective in increasing inference accuracy because they are collected from the same residential zone of test samples and customize the model for the use scenario. However, we do not observe such gains in the front-view cases, where the F1 score is even lower with our training. We gauge that it is likely because our training for the front-view one is improper without sufficient training samples (only 10 SFHs used to collect front views). It implies that good training is effective, but is not easy. More importantly, in this work, we do not intend to use customized training to improve

SS. *Second, training cannot fix all the errors, despite its effectiveness.* We clearly see that the roof is still misclassified as the pavement in two out of six SFHs (marked by ×, Figure 3a), which could misguide the drone to land on the roof, as the pavement areas close to the roof are easily treated as ideal landing sites. It is even worse in the front-view cases. The garage door is often misclassified as the pavement, which makes it hard to find the right/proper D-point; It even increases the risk of accidental collisions despite of built-in obstacle avoidance. *Third,* SS *is slow.* The average time needed to run SS once takes about 5.9 seconds (S23) and 2.1 seconds (Colab). It implies that SS likely fails to pace up with the drone's flight capability (up to a speed of 3 - 10 m/s); The drone has to land very slowly or cannot support autonomous landing in real-time (say, hovering until SS completes for the sake of safety and landing control).

Insights for SSS. Our design of SSS is driven by one heuristics: *common house structure can be exploited to enhance the poor performance of the legacy* SS*, which is unaware of the context of drone delivery to SFHs.* Specifically, there are three design insights: *First, there is no need for* SS *to recognize all the classes correctly for drone delivery.* Instead, we need to support very few classes of our interest. For instance, only two classes (roof and pavement) are needed with a clear top-view. When the view is obstructed by a tree, we need to consider two extra classes: tree and lawn. *Second, we even do not need to identify the complete shape of segments, namely, all the pixels.* Instead, what truly matters is the boundary of the interested classes, such as the boundary between the roof and driveway indicating the garage location, as well as the boundary between the roof and walkway denoting the front door as shown in Figure 3c. *Third, house structure does not change when the drone drops and changes its view.* It can be used to streamline SS across multiple continuous views. Moreover, it can be leveraged for movement detection, which is good for safety.

3 HOUSE-AWARE SEMANTICS OVER SSS

We devise SSS to leverage the SFH structure for autonomous door delivery (Figure 5). Evidently, in the the last-hundred feet scenario, the drone needs to handle three views (top-view, front-view and any-view) to determine and update the D-point. SSS enables house-aware semantics over common structural patterns in SFHs that follow the residential construction code (e.g., [1] for Indiana and similar codes in other states in the USA). In this work, we target at SFHs in the USA, which follow common structural patterns [4]:

- A SFH has a driveway which connects its garage to main road[1].
- A SFH usually has a walkway which connects the front door to the driveway or the main road.

As a result, they are converted into the following adjacent positional relationships while performing house-aware SS:

- When viewed from the top, (1) one roof means a SFH[2], (2) the roof is physically adjacent to one driveway (at the garage door), although the view might be fully or partly obstructed (see Figure 5), (3) the driveway is of any shape which primarily resemble rectangles or quadrilaterals with the length generally greater than the width.

[1]It is not true for a SFH with a detached garage or without a garage. But such SFHs are not common (not observed in our study) and we leave such SFHs as our future work.
[2]It is true in our study though it is not true for a SFH with a detached garage or other detached parts. For these SFHs, we will focus on the primary roof which covers the main dwelling parts.

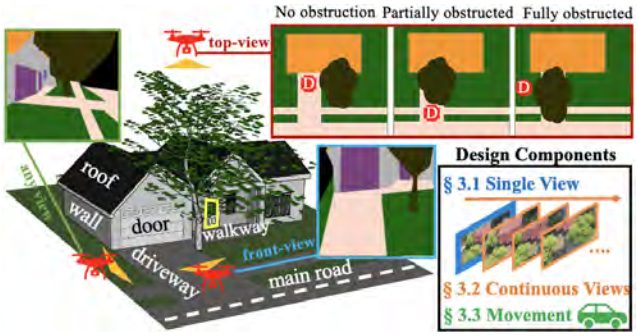


Figure 5: An overview of the design of SSS.

(4) the walkway, if applicable, is physically adjacent to the roof (the SFH) on one end and to the driveway or the main road on the other end, though the view might be fully or partly obstructed.

- In the front view or any view particularly when the drone descends low enough (say a few feet above the ground), (1) all the above SFH structure and the resulting positional relationships still hold, (2) the garage/front door and the walls, which are part of the roof (actually below the roof) in the top-view, become visible.

In a nutshell, the SFH structure and the adjacent positional relationships of segments of our interests (here, roof, pavement for driveway and walkway) are fixed in the physical world. We leverage this fact to enhance visual processing to locate the segments needed to determine the D-point, even though they are not fully visible in any view taken by the drone.

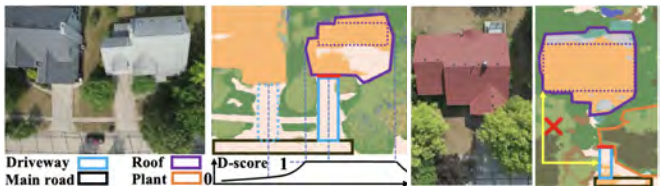
We notice that the initial D-point might be not good due to the limited field of aerial view (see the example in §1). Therefore, the D-point must be continuously adjusted based on the changing position and shooting angles of the drone. Given the constantly changing views, the unchanging house structure serves as a reference to accelerate visual processing across multiple continuous views.

SSS puts all above ideas together with three key modules: (§3.1) SSS over a single view to find or update the D-point, (§3.2) SSS across multiple views to quickly verify wether the D-point needs to be updated, and (§3.3) a light-weight movement check over a small region-of-interest to ensure a safe landing.

3.1 SSS for a Single View

There are two key insights of SSS to find the D-point: First, we are concerned with specific key structural elements (roof, driveway, walkway) rather than the entire segments. Second, based on the adjacency of these key components, we can swiftly identify a suitable D-point, away from the boundary, maintaining a certain safety distance (here, 1.5 m). However, if we observe that the extracted driveway and roof are not adjacent in the top-view, it may indicate potential obstructions. In such cases, we rely on other structures like trees or lawns to find a suitable boundary. As the drone descends, we then continuously track the boundary and update the D-point, which will be elaborated in §3.2. In the following, we will mainly use top-view to discuss how SSS works.

Top-view. The most important task is to identify roof and driveway, which assists the D-point selection. In order to fully utilize the SFH structure, we ensure the orientation of the orientation of drone is to face the target house, with the main road positioned at the bottom of the view. The details are elaborated at the end.



(a) Clear view (b) Obstrcuted view

Figure 6: Top-view over two examples.

However, this task is still challenging. As discussed in §2, the existing SS models perform poorly and they treat both public roads and private roads as pavement; Consequently, they cannot distinguish different types of pavement like driveway, walkway and main roads. To tackle this issue, we use the output of the legacy SS to further extract the main road and the roof based on their shapes. By this means, we limit the search range for the driveway if visible.

Algorithm 1 shows how to find the driveway and determine the D-point. We first extract the main road by locating the largest rectangle in the bottom third of the SS output with the labeled pavement. All classes detected as vehicles and obstacles will be replaced as pavement. Subsequently, we locate the initial roof by seeking the largest rectangle located in the middle of the view while considering all the parts classified as pavement and roof due to potential confusion by SS models (line 1). Note that not all roofs have rectangular shapes as shown in Figure 6, we utilize pixel similarity over YUV space to expand the detected rectangle (marked in purple dotted line), resulting in a complete roof (marked in purple solid line). After that, the region between the extracted main road and roof are the search range for the driveway (line 5-12). We implement a scoring mechanism to assess candidates within the search range (line 7). And choose the one with the highest score based on its shape, area, and *D-score*, which measures the relative distance from the extracted roof using a trapezoidal score function as shown in Figure 6a.

For the found driveway, we first verify whether it meets the standard (line 13). In cases where no suitable driveway is found, it may indicate that the driveway is either entirely concealed by trees or does not exist. Consequently, the drone will land above the lawn (line 14) and capture new views (e.g. any-view) to update the D-point. Otherwise, we further examine if the driveway intersects with the roof (lines 14-17). If it does (Figure 6a), it indicates the location of the garage door. The D-point is naturally selected at a safe distance from the boundary, preferably aligning with the center of the door. If there is no intersection, it suggests there is an obstruction in between (Figure 6b). We then select a location with the safety distance below the boundary line between the obstruction (e.g. trees) and the driveway as the D-point.

Note that we perform an extra adjustment when the starting point is insufficient to capture the entire house upon the drone's arrival at the target SFH. The drone initiates a random flight to find a viewpoint that can encompassthe entire house. In the meanwhile, we apply canny edge detection [12] and Hough Line Transform [17] to ensure that our captured view of the house is aligned properly. If not, a necessary rotation is made for adjustment.

Front-view and any-view. As the drone descends, front-views and any-views become available. Considering the limited effectiveness of SS in these views, we prioritize leveraging the drone's sensing data to perform a geometric transformation, involving rotating and projecting the semantic results onto a top view. We apply the above same way to get the boundary and update D-point. It's important to note that the boundary will now be between either the wall/roof and the pavement (driveway/walkway). However, performing SS for every captured view is resource-intensive and not feasible in real-time. Moreover, for different obstruction scenarios determined in Algorithm 1, we need to guide the drone to take different actions. We will handle these issues next.

Algorithm 1 Find Driveway and D-point

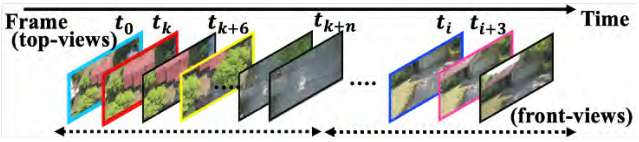
Input: Original image (O), semantic result (S = SS(O)), the maximum iteration count (N_{max}), score threshold (θ)
Output: D-point, Driveway ($BDrive$), obstruction type (Obs)

1: $Main$ = FindMainRoad(S), $Roof$ = FindRoughRoof(S)
2: $Roof$ = RefineRoof($Roof$,O)
3: $PaveArea$ = Crop(S == pavement ∪ S == gravel, $Main$)
4: $Score_{max} = 0$
5: **for** $n \leftarrow 1$ to N_{max} **do**
6: $Drive$ = FindLargestSqaure($PaveArea$)
7: $Score$ = $D\text{-}score(Drive, Roof)$ * Size($Drive$) * CheckShape($Drive$)
8: **if** $Score_{max} < Score$ **then**
9: $Score_{max} = Score, BDrive = Drive$
10: **end if**
11: $PaveArea$ = Crop($PaveArea, Drive$)
12: **end for**
13: **if** $Score_{max} < \theta$ **then**
14: D-point = FindD(S == grass, $Roof$), Obs = Fully Obstructed or No Driveway
15: **else if** CheckConnectivity($BDrive, Roof$) == $False$ **then**
16: D-point = FindD($BDrive, Roof$), Obs = Partially Obstructed
17: **else**
18: D-point = FindD($BDrive, Roof$), Obs = No Obstruction
19: **end if**
20: **return** D-point, $BDrive$, Obs

3.2 SSS Across Continuous Views

While the drone flies toward the initial D-point determined in §3.1, we continuously seeks better alternatives as new perspectives arise. As the house's physical layout remains stable, we can efficiently track the boundaries identified in §3.1 across multiple views and adjust the D-point accordingly. In Figure 7a, two types of continuous views are depicted: First, the views are captured when the drone is descending (t_0 to t_{k+n}) or flying horizontally without changing the camera angle. Second, the drone remains stationary while its camera angle keeps changing (after t_{k+n}). We will illustrate how we facilitate lightweight boundary checks for these views to update D-point. Furthermore, we will explain how SSS addresses invisible views, a unique case within continuous views.

Fixed camera angle. While descending towards the initial D-point, we maintain a downward-facing camera perspective to check if the identified boundaries are still valid. To achieve this, we can easily perform matrix transformations to zoom, rotate and match the current view with the previous one, with the assistance of the drone sensing data (e.g. GPS, altitude). As we have already identified the boundaries in the top-views, continuously matching enable us to



(a) Views over time

(b) Fixed camera angle (c) Changing camera angle

Figure 7: SSS over multiple views.

roughly know its position in the current view. We select t_k and t_{k+6} to show the matching in Figure 7b. It is evident that the matching is successful; all views align closely, with only slight mismatches due to drone vibrations. In summary, the descent process is akin to zooming in on a top view. Furthermore, we can swiftly refine and track the boundaries by inspecting specific regions containing essential house structures (such as the extracted driveway and roof), rather than analyzing the entire view.

Changing camera angle. However, when the drone is below a certain altitude, as the last view (t_{k+n}), the boundary between the tree and pavement may be out of sight. In such cases, adjusting the drone's camera angle to capture new house views becomes necessary. As the camera angle continuously adjusts, moving θ degrees upward in pitch, the views transition from top-views to front-views, as shown in Figure 7c. Utilizing known camera parameters and drone's orientation sensing data, we establish point correspondences between these two views via camera projection model [16]. This process enables real-time tracking of boundaries. When we have confidence in the results, such as when no obstructions hinder the top-views and SSS accurately recognizes the house structure, which can be inferred by finding a boundary of roof and driveway in the processed top-view, we may bypass SS for a lightweight check. In situations of low confidence, such as in cases of invisible views, we will handle them differently as follows.

Invisible views. After applying SSS for top-views (§3.1), we can determine the presence of obstructions. For example, in Figure 7, the driveway is partially blocked since the detected boundary is between driveway and tree instead of roof. As the drone descends below a certain altitude, like when it is under a tree, drone should adjust the gimbal pitch angle to let the camera face the house. We may apply SSS for the front-view, as introduced in §3.1, to acquire more semantic details about the house. Then we can detect the boundaries between the wall and the driveway. Additionally, knowing that the driveway is currently beneath the drone and should be visible from the bottom of the captured view, the localization becomes more efficient compared to the similar process used for top-views. In fully obstructed scenarios, the drone will take an additional step to figure out the location of the driveway with the any-view, focusing on the potential area between the identified roof and the main road. If it's confirmed that the driveway doesn't exist, the drone opts to land on the lawn instead.

3.3 SSS for Moving Objects

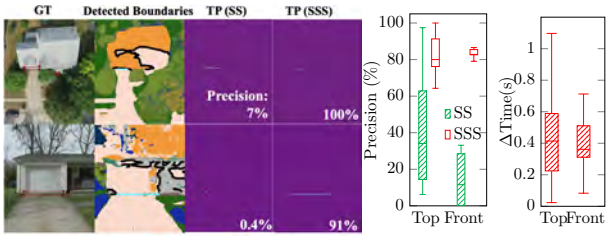
At the stage when the final D-point is determined, the landing process begins. To ensure the drone's safety, monitoring moving obstacles is essential. We devise sparse optical flow+ to detect motion and track the movement of potential obstacles (e.g. vehicles, dogs and pedestrians). Instead of performing pixel-wise matching for motion vectors between each pair of continuous frames, we focus on tracking a limited set of key points, which mainly located at the corners of objects. However, this approach has limitations. By concentrating on fixed key points, we may overlook newly appearing moving objects. Therefore, we need to update our key points wisely. For each frame, we aim to select new key points out of objects that have already been recognized and filter out previous noise points. Moreover, in some scenarios, there may be more than one moving object in the scene. To differentiate multiple moving objects, we leverage DBSCAN [26] to group motion vectors based on their speed and directions. Upon them a table is built to track each moving object and to predict future movements. Despite the drone being in motion, we apply optical flow to the matched view pairs, as discussed in §3.2, to eliminate its influence on misleading optical flow to find wrong motions. Moreover, SSS leverages the extracted driveway and walkway as RoIs and solely monitors them. If a moving object halts within the designated landing area as we approach the ground, we consider updating the D-point accordingly.

4 PRELIMINARY EVALUATION

We have implemented SSS into an Android app which controls the paired DJI drone for a waypoint-based flight, using Java (~3,600 lines of code) and Python (~1,900 lines of code). The core algorithm is implemented in Python; It acquires real-time video and sensor data through DJI Mobile SDK [13], updates the D-point over SSS and control the flight towards the updated D-point accordingly.

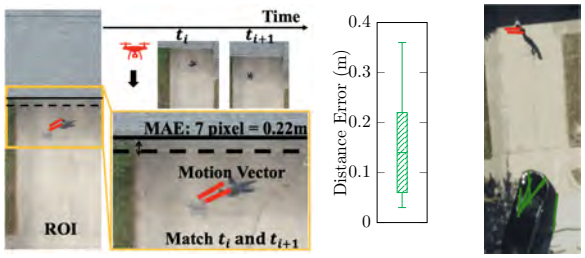
Experimental setting. We run a field test with two drone kits (Phantom 4 Pro V2 and Air 2S), which are tethered to a smartphone (Samsung Galaxy S23 Ultra) that implements SSS. We run drop-to-door experiments with 10 participating SFHs (Figure 2c), which are located in one residential zone (Figure 2b). These target SFHs are divided into two groups: garage door delivery (labeled from A to D) and front door delivery (labeled from E to J), based on whether the walkway is safe to fly. In this study, we set the safety distance from an obstacle on the front as 5 ft (1.5 m) and the one from the side as 1.5 ft (0.5 m). The experiment starts at one point at an altitude of 100 – 150 ft (30 – 45 m) above the target SFH.

Microbenchmark: Single view. We achieve almost a two-fold accuracy improvement of SSS over SS for a single view with no more than 8% computing overhead. Figure 8a uses the same example instance from Figure 3 and the outcome of SSS and SS is labeled in cyan and black respectively (bolded the results for better visibility). To quantify the accuracy improvement, we introduce a new metric: boundary precision, instead of using pixel-wise classification accuracy. $Precision_{boundary} = TP/bound_{detected}$, where TP is $bound_{labeled} \cap bound_{detected}$. It is challenging to achieve a perfect boundary without pixel deviations by labeling. To compensate for it, we expand the labeled boundary by *delta* pixels, representing a real-world distance of 0.2 meters. The specific *delta* values differ based on views (here, 7 pixels for top-view and 25



(a) SSS over top-view and front-view (b) Precision (c) Δ Time

Figure 8: SSS vs. SS for a single view.



(a) Example Instance (b) DE (c) MO

Figure 9: SSS over multiple views and moving objects (MO).

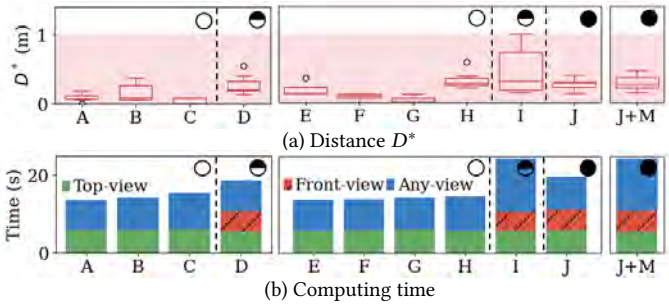
pixels for front-view). We evaluate the precision over 10 selected SFHs (Figure 8b), SSS outperforms SS significantly, particularly in the challenging front-view (averagely 84% over 16%). In terms of increasing computing time (Figure 8c), on average, SSS only adds 430ms to the processing time compared to SS for both views.

Microbenchmark: Multiple views and movement. We use the distance error (DE) to evaluate the accuracy of boundary tracking across multiple views. Figure 9a shows two captured adjacent views (with a 1s interval) in the drone descent and the matched results. The black dashed line represents the found boundary in the lower view, which differs by 7 pixels (namely, 0.22 m in this instance). Figure 9b shows that the overall DE is small (mostly below 0.22 m and all below 0.4 m). Meanwhile, Figure 9c shows the output of sparse optical flow+, revealing the successful detection of both moving objects. Boundary tracking and sparse optical flow+ are both lightweight, taking about 190 ms and 10 ms, respectively.

Drop-to-Door Performance. We evaluate how SSS assists the door delivery over 10 SFHs with different drop-off complexities: SFHs D, I, J do not have fully visible driveways from the above, which are obstructed by a tall tree. Among them, the top-view of J cannot see the driveway at all. In addition to the static scenario, we take into account the presence of moving cars for SFH J, denoted as J+M. For each setting, we perform at least 10 runs.

∘ *Precision.* We use the distance (D^*) of actual D-point from the best D-point (1.5m either from the front door or the garage) to quantify the accuracy. We observe, in all the experiments (Figure 10a), drone successfully drops near the best drop-off point with D^*<1 m. The median D^* is below 0.5 m. Especially noteworthy, even in situations where the D-point is entirely out of view from the top-view (case J, J+M), we can still ensure a safe landing at the door. This underscores the effectiveness of SSS.

∘ *Computing Overhead.* We present the total computing time for SSS on different views in Figure 10b. We significantly reduced the number of times we need to invoke semantic segmentation. For all test cases, it requires a maximum of only four calls (SFH



(a) Distance D^*

(b) Computing time

Figure 10: Performance of SSS for 11 test cases.

J), with the majority requiring only two calls. The frequency of calls depends on the visibility of the driveway in the front view. When the driveway is partially or fully obstructed, we need to run SSS over more views. For instance, in SFH J, we apply SSS in the top-view, front-view and two any-views. Specifically, over the top-view, D-point will be initialized over a lawn, giving no driveway can be seen. Then observe the SFH from the lawn to determine the driveway's location. Upon reaching the driveway, the perspective shifts to face the front, searching for the walkway. Finally, we follow the walkway to find the front door. Besides, comparing the J+M with J, we find the processing time is slightly larger, showing the low computing overhead for both static and moving scenarios. In a nutshell, for all SFHs, the computing can be done within 25 s.

5 RELATED WORK

One most relevant work is [19], which also targets at dropping off at the door using SS. However, it is based on simulations and many practical problems are largely overlooked; It focuses on classifying the front and back yard primarily based on the house orientation, heavily relying on the pixel-level performance of the SS model, without exploiting the known and common house structure, which is the focus of our work. A few studies [11, 14, 18, 20] also use SS to solve a similar problem (landing) but they are all based on theoretic models or simulations. Other studies [7–9, 22, 27] are centered on the last-mile problem instead of the last-hundred feet landing problem. To our best knowledge, there is no prior work in the literature that exploits a house structure to expedite and improve semantic segmentation for drone landing towards the door.

6 CONCLUSION

We present our preliminary efforts towards autonomous drone delivery to the door. In this work, we develop structural semantic segmentation (SSS), a new computer vision technique that leverages known house structure to efficiently decide the drop-off points so as to control the landing flight of the drone in real-time. While promising, there are many remaining issues including but not limited to handling SFHs that do not follow our basic structural rules as well as other residential houses like condos, apartment and even multi-floor buildings. The developed SSS is sensitive to light conditions and performs poor at nighttime, which calls for an enhanced technique that works robustly well under various conditions.

Acknowledgements. We thank all anonymous reviewers for their insightful comments. The work has been partially supported by NSF grant CNS-1750953.

REFERENCES

[1] 2020. 2020 Indiana Residential Code (675 IAC 14-4.4). https://www.in.gov/dhs/files/675-IAC-14-4.4-Indiana-Residential-Code-Proposed-Draft-Rule-LSA-Formatted.pdf. (2020).
[2] 2022. Multiclass Semantic Segmentation of Drone Images Using DL. https://github.com/ayushdabra/drone-images-semantic-segmentation. (2022).
[3] 2022. Semantic Drone Dataset. http://dronedataset.icg.tugraz.at/. (2022).
[4] 2022. U.S. single family homes - statistics & facts. https://www.statista.com/topics/5144/single-family-homes-in-the-us/#topicOverview/. (2022).
[5] 2024. Demo for Autonomous Drone Delivery to Your Door. https://www.youtube.com/playlist?list=PLLzN69of2f9bD5ZfPYhDpkAv-Pz2zI7Ac. (2024).
[6] 2024. Release for DroneDelivery (HotMobile'24). https://github.com/mssn/dd-demo. (2024).
[7] Fabio Borghetti, Claudia Caballini, Angela Carboni, Gaia Grossato, Roberto Maja, and Benedetto Barabino. 2022. The Use of Drones for Last-Mile Delivery: A Numerical Case Study in Milan, Italy. *Sustainability* 14, 3 (2022).
[8] Nils Boysen, Dirk Briskorn, Stefan Fedtke, and Stefan Schwerdfeger. 2018. Drone Delivery from Trucks: Drone Scheduling for Given Truck Routes. *Networks* 72, 4 (2018), 506–527.
[9] Gino Brunner, Bence Szebedy, Simon Tanner, and Roger Wattenhofer. 2019. The urban last mile problem: Autonomous drone delivery to your balcony. In *2019 international conference on unmanned aircraft systems (icuas)*. IEEE, 1005–1012.
[10] McKinsey & Company. 2023. Global drone logistics market in 2023. https://www.globalialogisticsnetwork.com/blog/2023/05/03/global-drone-logistics-market-in-2023/. (May 2023).
[11] Harsimret Singh Dhami, Dmitry Ignatyev, and Antonios Tsourdos. 2022. Semantic Segmentation based Mapping systems for the Safe and Precise Landing of Flying Vehicles. *IFAC-PapersOnLine* 55, 22 (2022), 310–315.
[12] Lijun Ding and Ardeshir Goshtasby. 2001. On the Canny edge detector. *Pattern recognition* 34, 3 (2001), 721–725.
[13] DJI. 2022. Mobile SDK. https://developer.dji.com/mobile-sdk/. (2022).
[14] Eitan Frachtenberg. 2019. Practical drone delivery. *Computer* 52, 12 (2019), 53–57.
[15] Google. 2022. Google Colaboratory. https://colab.google/. (2022).
[16] Junpeng Guo and Chunyi Peng. 2021. Towards live video analytics with on-drone deeper-yet-compatible compression. *arXiv preprint arXiv:2111.06263* (2021).
[17] John Illingworth and Josef Kittler. 1988. A survey of the Hough transform. *Computer vision, graphics, and image processing* 44, 1 (1988), 87–116.
[18] Efstratios Kakaletsis, Charalampos Symeonidis, Maria Tzelepi, Ioannis Mademlis, Anastasios Tefas, Nikos Nikolaidis, and Ioannis Pitas. 2021. Computer Vision for Autonomous UAV Flight Safety: An Overview and a Vision-based Safe Landing Pipeline Example. *ACM Computing Surveys (CSUR)* 54, 9 (2021), 1–37.
[19] Shyam Sundar Kannan and Byung-Cheol Min. 2022. Autonomous Drone Delivery to Your Door and Yard. In *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 452–461.
[20] Joe Kinahan and Alan F Smeaton. 2021. Image Segmentation to Identify Safe Landing Zones for Unmanned Aerial Vehicles. *arXiv:2111.14557* (2021).
[21] Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully Convolutional Networks for Semantic Segmentation. In *CVPR*.
[22] Victor RF Miranda and et al. 2022. Autonomous navigation system for a delivery drone. *J. Control. Autom. Electr. Syst* 33, 1 (2022), 141–155.
[23] Ecommerce Next. 2023. Top 10 Drone Delivery Companies. https://www.ecommercenext.org/top-10-commercial-drone-delivery-companies/. (Jan 2023).
[24] Market Research. 2022. Drone Package Delivery Market: Global Forecast to 2030. https://www.marketsandmarkets.com/Market-Reports/drone-package-delivery-market-10580366.html. (July 2022).
[25] Market Research. 2023. Drone Package Delivery Global Market Report 2023. https://www.researchandmarkets.com/reports/5769463/drone-package-delivery-global-market-report. (April 2023).
[26] Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. DBSCAN revisited, revisited: why and how you should (still) use DBSCAN. *ACM Transactions on Database Systems (TODS)* 42, 3 (2017), 1–21.
[27] Judy Scott and Carlton Scott. 2017. Drone delivery models for healthcare. In *Proceedings of the 50th Hawaii International Conference on System Sciences*.
[28] Richard Szeliski. 2022. *Computer vision: algorithms and applications*. Springer Nature.