DOI: 10.1111/cgf.14668
Pacific Graphics 2022
N. Umetani, E. Vouga, and C. Wojtan
(Guest Editors)

Volume 41 (2022), Number 7

A Drone Video Clip Dataset and its Applications in Automated Cinematography

Amirsaman Ashtari[1], Raehyuk Jung[1], Mingxiao Li[2], and Junyong Noh[1]

[1]KAIST, Daejeon, South Korea
[2]University of Waterloo, Waterloo, Canada

Abstract

Drones became popular video capturing tools. Drone videos in the wild are first captured and then edited by humans to contain aesthetically pleasing camera motions and scenes. Therefore, edited drone videos have extremely useful information for cinematography and for applications such as camera path planning to capture aesthetically pleasing shots. To design intelligent camera path planners, learning drone camera motions from these edited videos is essential. However, first, this requires to filter drone clips and extract their camera motions out of these edited videos that commonly contain both drone and non-drone content. Moreover, existing video search engines return the whole edited video as a semantic search result and cannot return only drone clips inside an edited video. To address this problem, we proposed the first approach that can automatically retrieve drone clips from an unlabeled video collection using high-level search queries, such as "drone clips captured outdoor in daytime from rural places". The retrieved clips also contain camera motions, camera view, and 3D reconstruction of a scene that can help develop intelligent camera path planners. To train our approach, we needed numerous examples of edited drone videos. To this end, we introduced the first large-scale dataset composed of edited drone videos. This dataset is also used for training and validating our drone video filtering algorithm. Both quantitative and qualitative evaluations have confirmed the validity of our method.

Keywords: cinematography, aerial videography, dataset, motion planning, quadrotor camera

1. Introduction

Thanks to their ability to freely move in 3D space, drones are opening new ways for acquiring videos. Over the last few years, they became popular video capturing tools in various contexts and therefore have been recently investigated in computer graphics and related fields. Applications include automated drone cinematography [JRT*15,RH16,NMD*17,GSH18,GLC*18,XYH*18, ASN*20,GHN*16,GH21,NAMD*17,BBS*21,GFTG16,HLY*19, JJG*16, GH18], human motion capture [NOP*18, ZLP*18], drone motion taxonomy [NMRB17], surveillance [SPO18], drone pursuit [ÇTM18], detection/identification [CKL*18, HLH17], 3D scanning [RSD*17, SMGH18], safe landing zone detection [KSI*19], disaster management [Res15], and journalism [TC14].

In the computer graphics and video indexing literature, no existing studies focus on *edited drone videos* in the wild, although they constitute the highest portion of drone videos on the Internet. A drone video created using editing programs (i.e., an *edited drone video*) is commonly composed of several *clips*, each of which is defined by scene cuts. Each clip is different from other clips in that edited video. For example, a clip can be a drone or non-drone clip or can be captured indoors or outdoors, as shown in Figure 2.

Once videos are captured, video editing is a common process [TBLA16, WYH*19]. *Edited drone videos* are first captured and then edited by humans to contain aesthetically pleasing drone camera motions and scenes. Therefore, they have extremely useful information for automated drone cinematography and for applications such as camera path planning to capture aesthetically pleasing shots [JWW*20,JCW*21]. To design intelligent camera path planners, learning drone camera motions from these edited videos is essential. To this end, we propose the first dataset that contains necessary data for training intelligent drone path planners (i.e., drone camera views and their corresponding camera paths). Construction of such a dataset is extremely hard because it requires to 1) decompose the edited drone videos into their individual clips, 2) detect drone clips among these individual clips that commonly contain both drone and non-drone clips, and 3) finally extract camera paths of the detected drone clips. Therefore, we focus on data construction in this paper for the benefit of related fields that require training of intelligent drone camera path planners.

None of the existing drone datasets contain edited drone videos. The underlying reason is that the construction of such a dataset requires a very time-consuming and cost-inefficient process of manually filtering drone video clips out of edited drone videos by defining cut times and clip types (i.e., drone vs. non-drone). Moreover,

Figure 1: *Representative frames of our DVCD18K dataset. Note the great diversity in terms of appearance, scene category, shot type, camera orientation, and time, among others.*

there is no method that can automatically distinguish the simple difference between a drone and non-drone video frame, nor a method to filter drone clips out of a video collection. In addition, there is no existing drone datasets to train a video filtering algorithm to return drone clips corresponding to a search query because none of the datasets provide labels such as drone/non-drone or cut times, required for training and evaluating a drone video clip filtering algorithm. Existing video search engines such as YouTube also return the whole edited video as a semantic search result and cannot return only drone clips inside an edited video.

While there are some existing studies for action recognition in drone videos, they are focused on surveillance or human activities. In our random drone video samples, we measured that only 4.51% of edited drone videos in the wild contain human faces and activities, and drones are mostly used to capture scenic aerial views. Focusing on human activity detection in drone videos would result in wasting a big percentage of drone videos shared on the Internet, whose purpose is to capture aesthetically pleasing aerial scenes.

Most existing drone datasets are composed of non-edited raw drone videos for object/human detection and tracking in aerial shots, as shown in Table 1. These datasets do not consider the aesthetically pleasing aspects of drone videos due to their targeted application such as object detection. Therefore, they cannot be used to train intelligent drone path planners. To train intelligent drone path planners, one requires information about drone camera paths and its corresponding camera view to imitate the way expert drone camera operators capture a scene. These path planners also require a 3D reconstruction of a scene to avoid obstacles in their algorithm design. Unfortunately, there is no drone dataset that focuses on extracting drone camera paths from videos.

One naive solution to detect drone clips and extract their camera path data is to first decompose the video collection into its individual clips using existing cut-detection methods (Section 4), and then identify the drone clips using CNN models that are trained to detect drone vs non-drone clips (Section 5). However, we observed that flaws of existing video cut detection methods adversely impact the precision of filtered drone clips.

To address the gaps mentioned above in the literature, we contribute the followings:

- We introduce DVCD18K: a large-scale, annotated Drone Video Clip Dataset (Section 3), which is the first dataset composed of 44 hours (18K clips) of "edited drone videos" (Figure 1). Our dataset contains various annotations such as the cut time, cut transition effect, clip category (drone/non-drone), scene category (urban/rural/mixed), location(indoor/outdoor), time (day/night/between), logo and text existence, as well as the information about 3D path and 3D reconstruction for drone clips. Our dataset is the first dataset containing drone camera motions, drone camera view, and 3D reconstruction of a scene that can help develop intelligent drone camera path planners.

- Leveraging our dataset, we proposed the first approach that filters and retrieves specific types of drone video clips from an unlabeled video collection using high-level search queries, such as "outdoor drone clips captured from rural places during the day" (Section 6), which was impossible using existing drone datasets. For this, we trained the first classifiers in the aerial video domain to detect drone/non-drone, scene category (urban/rural/mixed), location (indoor/outdoor), time (day/night/between), and logo presence in aerial shots (Section 5), which was not possible using the labels of existing drone datasets or even ground video datasets (e.g., for drone/non-drone, and urban/rural categories).

- To address flaws of existing video cut detection methods (Section 4), we proposed our own drone content-aware cut detection method (drone frame block detection) in Section 6.1. We quantitatively proved that our method improves the precision and duration of filtered drone clips in comparison with four out-of-the-box video cut-detection methods in detecting true drone clips (Section 6.3).

- We developed the first method to automatically filter drone clips with specific drone camera motion patterns such as drone shots captured by circular or linear (backward/forward) camera paths (Section 6.3), which confirms the validity of our extracted paths.

Table 1: *Summary of existing datasets.*

Dataset name	No. videos	No. clips	Duration	Purpose	Content	Edited videos
UG^2 [SVB*20]	629	629	32.7 hours	visual enhancement and object detection	captured by UAVs, gliders, and on-ground cameras	No
Professional drone human motion [HLY*19]	92	92	38 minutes	Learning to film from human motion videos	6 filming styles of human motions (e.g., walking)	No
HighD [KBKE18]	60	60	16.5 hours	safety validation of automated vehicles	drone videos only captured on German highways	No
UAVDT [DQY*18]	100	100	44 minutes	object detection and tracking	captured at a number of locations in urban areas	No
VisDrone [ZWB*18]	263	263	1.7 hours	object detection and tracking	contains different cities in China	No
Okutama-Action [BMS*17]	43	43	43 minutes	human action detection	captured at a baseball field in Okutama, Japan	No
CARPK [HLH17]	7	7	24 minutes	object counting and object localizing	contains 4 different parking lots	No
UAV123 [MSG16]	123	123	59 minutes	object detection and tracking	captured mostly by low-altitude UAVs	No
Stanford dataset [RSAS16]	60	60	4.9 hours	trajectory forecasting and target tracking	captured by static drones in Stanford campus	No
Mini-drone [BKRE15]	38	38	13 minutes	study of privacy in video surveillance	captured in parking lots	No
DVCD18K (ours)	**991**	**18,551**	**44.3 hours**	**semantic drone video search to provide data for training intelligent drone path planners**	**contains various *edited* videos, countries, locations, low/high altitudes, time, scene types and labels**	**Yes**

2. Review of existing datasets

In the following, we review existing drone video datasets while focusing on the number of videos, clips, duration, scene types, purpose, and annotations. The highD dataset [KBKE18] contains drone videos captured only on German highways. It is composed of 60 video clips with a total duration of 16.5 hours, and the vehicles are annotated with bounding boxes, vehicle class, driving direction, and mean speed. The Okutama-Action dataset [BMS*17] focuses on human action detection and includes 43 video clips with a total duration of about 43 minutes. These clips are captured by drones at a baseball field located in Okutama, Japan, and each clip is annotated with bounding boxes of people and labels of their corresponding action. Videos in both the highD [KBKE18] and Okutama-Action [BMS*17] datasets are captured in only one specific kind of location. The dataset from Huang et al. [HLY*19] contains 92 video clips for a total duration of 38 minutes. The videos are labelled and grouped into 6 filming styles based on the camera motion but are limited to human motion scenes.

The VisDrone dataset [ZWB*18] consists of 263 video clips with a total duration of around 1.7 hours[†]. The videos are manually annotated with bounding boxes of targets of interests, such as bicycles, cars, and pedestrians and are captured in different locations (different cities in China) and environments (urban and country). The UAV123 dataset [MSG16] contains 123 video clips with a total duration of 59 minutes. The videos are mostly captured from low-altitude UAVs and annotated with bounding boxes of targets of interest such as bikes, birds, and boats. The UG^2 dataset [SVB*20] contains 629 videos with a total duration of around 32.7 hours[†] taken by UAVs and manned gliders, as well as videos taken on the ground. The UG^2 annotations provide bounding boxes establishing object regions and classes for the purpose of object classification, detection, tracking, and video enhancement. The UAVDT dataset [DQY*18] contains 100 video clips in urban environments and has a total duration of 44 minutes[†]. This dataset is annotated with bounding boxes for cars. The CARPK dataset [HLH17] comprises 7 video clips of drone views with a total duration of 24 minutes. The dataset focuses on car counting and provides annotations of bounding boxes for cars. The Stanford drone dataset [RSAS16] contains 60 video clips with a total duration of approximately 4.9 hours and is dedicated to trajectory forecasting and target tracking. Targets of interest such as pedestrians, skateboarders, and bicyclists are annotated with bounding boxes along with their class label and trajectory. The video clips are captured only by static drones hovering in the air. The mini-drone dataset [BKRE15] is dedicated to the study of privacy in video surveillance. It provides 38 video clips with a total duration of 13 minutes, and each vehicle and person are annotated with bounding boxes.

Overall, the existing datasets are only composed of individual raw clips (i.e., no *edited* videos), mostly have a relatively small size both in terms of duration and number of clips, and/or have a limited variety, for example in targeted application, camera altitude, location, and scene type.

Summary and comparison: The complete summary of the existing drone video datasets is available in Table 1. Our dataset provides four major differences with respect to existing datasets: 1) Our DVCD18K offers a significantly larger scale in terms of number of videos, number of clips, and duration. 2) It offers a great variety of drone videos in terms of location, motion, appearance, camera altitude, scene category, shot type, quality, and editing effect, as shown in Figure 1. 3) Instead of just individual raw videos, it contains *edited* drone videos, for example with video cuts, video effects, and combination of drone/non-drone clips. 4) It provides various levels of annotation, such as manual ground truth information on scene description, video editing, social platform metadata, as well as computed annotation (e.g., camera path by SLAM and 3D scene reconstruction). Our dataset is the first large-scale, fully annotated dataset of *edited* drone videos. As will be detailed in the following, this brings new challenges, opportunities, and applications such as semantic drone video clip search to provide necessary data for training intelligent drone path planners.

[†] The duration of annotated data was not available, so we calculated it from the known number of frames assuming a framerate of 30FPS.

3. A Comprehensive Dataset of Drone Videos

3.1. Overview of our DVCD18K dataset

Leveraging the great amount of data available on the Internet, we automatically gather a large number of drone videos from websites. In our early experiments, we were planning to use videos from Youtube by searching for “drone videos" and similar queries, but it returned a very high percentage of non-drone videos, for example people talking about drones. To solve this issue, we privileged video platforms dedicated to drone videos, and selected AIRVŪZ[‡] “the world’s leading drone video sharing platform".

We annotated 991 drone videos randomly selected from the video platform. These videos are at the core of our dataset, and we used them as the source videos for the following annotation work and video analysis. Annotators were asked to first decompose each video into individual clips and then annotate each clip. On average, for one hour of source video, it took 5 hours of manual annotation and 9.3 hours of annotation computation, e.g., path computation by SLAM. The annotation of the whole dataset took around 630 hours, including 220 man-hours of manual annotation and 410 hours of computed annotation. An illustration of our dataset annotation is available in Figure 2, and details about the annotation are provided in Section 3.2 and 3.3. The dataset statistics are provided in Section 3.4. The supplementary material contains our dataset license, link, dataset documentation via “Datasheets for Datasets" [GMV*18] framework, and our codes.

3.2. Manual annotation

Overall procedure: Our dataset consists of 991 source videos, and each of them is composed of a number of clips, which could be either drone or non-drone. The goal is to identify all the drone video clips and annotate them. For this, our overall procedure proceeds as follows: given a source video, we first asked annotators to decompose it into clips by annotating the shot boundary times in the form of start/end times. The annotators then identified which of the resulting clips are captured by drones. Finally, for each identified drone video clip, we asked the annotators for extra annotations such as indoor/outdoor.

Procedure details: The annotators were provided with clear instructions about the labelling. In order to remove any potential confusion, we gave the annotators descriptive requirements with representative examples. To have a common annotation environment, the annotators were told to use the same online player on AIRVŪZ website and report the video clip time in seconds which is the precision granularity of the video player. Furthermore, for each clip boundary, we asked the annotators to label the type of clip transition such as hard cut, fade-in/out, and zoom-in/out.

The main focus of our DVCD18K dataset is on drone video clips. Originally, we were planning to separate our dataset clips into just two classes: drone vs. non-drone. While it sounded intuitive, we found out that it was not entirely satisfactory for two main reasons. The first reason is motivated by the fact that our dataset could be used for drone motion-based applications, such as automatic drone motion planning. However, the apparent drone motion could be affected by video editing, such as the rewinding effect[§]. Therefore, we decided to divide the drone clips into two subcategories: drone and droneE, which respectively represents the clip without and with video editing effects that can affect the drone camera path interpretation. The second reason is from the observation that several videos contain “strobed" content or series of rapidly flickering extremely short clips/images, typically consist of around 10-15 shots that are shown for less than 0.5 seconds. This strobe effect is a popular video editing technique for creating a sense of dynamism and excitement in the video. However, in practice, when searching videos over the Internet, users may not be interested in such ultra short individual clips. Moreover, the most time-consuming annotation task is to identify the video cuts and report their times. Therefore, we added a class named strobe, which contains the series instead of the individual clips of the series. In summary, the annotators classified each clip of the dataset into one of the following four categories:

- drone: drone clips without any video editing effects that may affect the interpretation of the camera paths. This category includes fast forward and slow motion.
- droneE: drone clips with editing effects that may affect the interpretation of the camera paths. This category includes rewinding and video shakes.
- nondrone: clips not captured by drones. This category includes hand-held videos, videos of people talking about drones, animations, and drawings.
- strobe: series of extremely short videos/images.

Subsequently, for each clip in the drone category, we asked the annotators to provide the following information (collectively referred to as semantic annotation in the remainder of this chapter): location (indoor or outdoor), time (daytime or nighttime or in-between like sunrise and sunset), scene type (urban or rural or mixed, i.e., mixture of rural and urban), whether it contains any logo and text, as well as the start and end transition effect.

Sometimes, it is hard for humans to determine whether or not a clip is captured by a drone. Consequently, we asked the annotators to report a score from 1 (not sure at all) to 10 (perfectly sure) for each drone and nondrone clip of the dataset as a confidence indicator. We also provide this confidence score in the dataset. Note that we annotate all drone clips regardless of their confidence scores.

Annotation validation: After annotation, we validated all the annotated results. One way would be to ask a second set of annotators to independently annotate the videos using the same procedure as described above and then compare the annotation results. However, the repeated process would be time consuming and not necessarily effective. We found that the most time consuming annotation task is to identify when a video cut happens and report its time. Therefore, in our annotation validation round, we first cut the source videos into each individual video clip based on the cut time information of the annotation sheet. These clips were then provided to a second set of annotators, who were asked to indicate whether

[‡] https://www.airvuz.com/

[§] Rewinding effect is a special effect in cinematography whereby the action that is filmed is shown backwards or time-reversed.

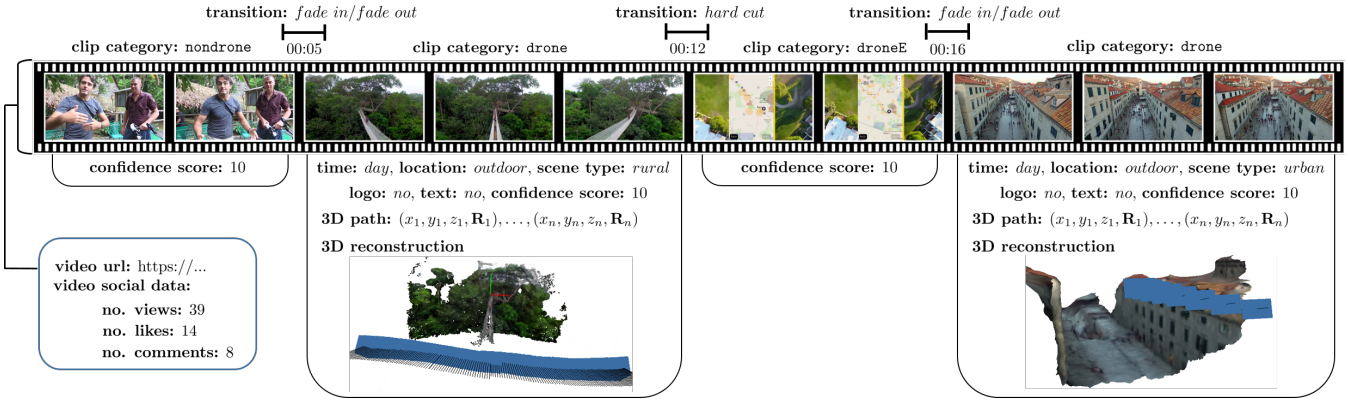


Figure 2: *Illustration of our dataset annotation. Our dataset is provided in a .CSV file with annotated labels such as the transition effect, cut time, location, scene type, drone/non-drone, logo and text existence as well as 3D path and 3D reconstruction for drone video clips.*

or not the clip contained a cut as well as label semantic annotation. Shot type and cut time information were validated for every clip. Additional semantic information such as scene type was re-annotated only for drone clips. If an error in cut time or conflicting information was reported, corresponding clips were revised according to the validation. This validation annotation was conducted in a "blind-way": the annotators did not have access to the other set of semantic annotations. If some annotations in the validation round did not coincide with the original annotations, a third set of annotators checked the corresponding parts in the source video and finalized the annotation.

3.3. Automatic annotation

Social platform metadata: The webpage of each video on the AIRVŪZ website contains social platform metadata that we also provide in our dataset. Metadata includes the number of views, likes, and comments.

Camera path and 3D reconstruction: For each drone video clip, we provide the 6D drone camera path (position and orientation) computed by both ORB-SLAM2 [MT17] and Metashape [Agi]. We also provide a 3D reconstruction of the scene by Metashape. The 3D reconstruction is available as both a 3D point cloud and a textured mesh. Representative results of the camera path estimation and 3D reconstruction from drone video clips are available in Figure 2 and Appendix 9.1. On average, it took 8.6 and 2.4 minutes to process 1 minute of input video at 640×480 resolution by Metashape and ORB-SLAM2, respectively. In total, it took around 320 and 90 hours of computation time for the path estimation and 3D reconstruction, respectively, for all the drone clips of our dataset. See Appendix 9.1 for details.

3.4. Dataset statistics

Our DVCD18K dataset contains a total of 18,551 drone video clips issued from 991 videos, and the total duration is approximately 44 hours (Figure 1). 85% duration of the clips belong to the drone category, 3% to droneE, 10% to nondrone and 2% strobe. The duration of each video spans from 17 seconds to 14.5 minutes with mean = 2.7 minutes and median = 2.5 minutes, as shown in Figure 3(a). It also shows that the great majority (81%) of the videos are less than 4-minutes long, and 90% are between 43 and 327 seconds.

Figure 3(b) plots the relation between the video duration and the number of clips. It shows that the number of clips has a dominant linear trend with respect to the video duration. This tends to suggest that video editors might decide the clip length independently of the total video duration. Figure 3(c) plots the distributions of the drone and non-drone clip duration. First, contrary to what we originally expected, it shows that their distributions are similar. This tends to indicate that video editors select clip length independently of the drone or non-drone contents. Second, the duration of each drone video clip spans from 1 to 281 seconds with mean = 7.53 seconds and median = 5 seconds. It also shows that the great majority (82.6%) of the drone video clips are less than 10-second long, and 90% are between 1 and 21 seconds. Furthermore, we analyze the popularity of videos based on their number of views, likes and comments. Figure 3(d) shows the distribution of the number of views. It shows that most of the videos (77.5%) are seen less than 3,000 times while the most top 4.8% popular ones are seen more than 9,500 times. See Appendix 9.3 for the number of likes and comments.

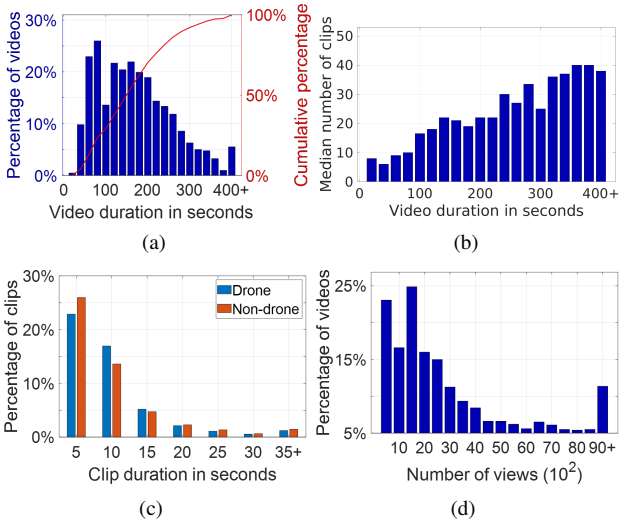


Figure 3: *(a): distribution of the video duration. (b): distribution of the number of clips per video with respect to video duration. (c): distribution of the duration of all the identified drone and non-drone video clips. The Y axis represents the percentage of clips per category normalized for each category. (d): distribution of the number of views.*

4. Baseline evaluations of video cut detection

One naive solution to detect drone clips is to first decompose the video collection into its individual clips via a cut detection method and then detect drone clips among these individual clips. In this section, we evaluate existing methods of video cut detection based on our manual ground truth annotation of clip time boundaries. We used the following methods for video cut detection:

ORB-SLAM2: In the context of stereo magnification, which enlarges the baseline of views captured by a narrow-baseline stereo camera, Zhou et al. [ZTF*18] presented a method based on ORB-SLAM2 [MT17] to automatically divide a video into individual clips. We follow their strategy: we feed successive video frames to ORB-SLAM2 to track the camera, consider the tracking failure as a video cut, and resume the algorithm to detect the next video cut. We use the default vocabulary file and camera calibration file.

Yahoo Hecate: Hecate is a video processing library developed by Yahoo Research [SRVJ16]. Shot boundary detection is one of its complementary functions and is achieved by computing the edge change ratio between two consecutive frames. We use the default parameter settings.

PySceneDetect: We evaluate the Python application PySceneDetect [Cas18]. Among the various methods available, we select the advanced content-aware cut detection algorithm and use the default hyperparameter setting.

ffprobe: We evaluate the shot boundary detection named ffprobe [FFm18]. It detects scene changes using scene filters. We set the ffprobe to report a cut when the estimated scene detection score computed based on the absolute differences is greater than a threshold. According to the ffprobe user manual, this threshold should be between 0.3 and 0.5, and we set it to 0.3. Here, the lower means more sensitive.

Evaluation: All cut detection methods were applied on the whole videos, but the evaluation metrics were computed only on drone video clips. Because we focus on drone clips, our annotators just provide the start and end time for the consecutive non-drone clips and do not provide manual ground truth of video cuts in between. In our evaluations, we allowed a 1-second margin between the detected cut times and the ground truth. For example, if a method detects a cut at 00:05, we would consider it as correct if its time from the annotation sheet is 00:05$\pm$1s. The margins were allowed because our ground truth manual annotations were reported in seconds.

To evaluate the cut detection methods, we follow a procedure similar to Krulikovska and Polec [KP12]. The results are available in Table 2. Overall, the F-score in Table 2 shows that PySceneDetect and ffprobe outperform the other two approaches and have approximately similar performance, e.g., high correctly detected cuts and high precision as well as low missed cuts and high recall. Hecate has the highest falsely detected cuts and lowest precision while ORB-SLAM2 has the highest missed cuts and lowest recall. Hecate is overly sensitive to changes of frames and consequently reported highest number of cuts 123,227 with the lowest precision. ORB-SLAM2 has the highest cut detection precision followed by ffprobe. While ORB-SLAM2 and ffprobe have higher precision in comparison with others, based on the recall score, ffprobe substantially outperforms ORB-SLAM2 by detecting a higher ratio of correct cuts among all ground truth cuts.

Table 2: *Evaluation of video cut detection methods. Correct stands for correctly detected cuts, Missed for missed cuts, and False for falsely detected cuts.*

	Precision	Recall	F-score	Correct	Missed	False
Hecate [SRVJ16]	0.15	0.90	0.25	18,156	1,988	105,071
PySceneDetect [Cas18]	0.62	0.69	0.65	13,832	6,312	8,670
ffprobe	0.71	0.57	0.63	11,394	8,750	4,548
ORB-SLAM2 [MT17]	0.83	0.16	0.26	3,159	16,985	644

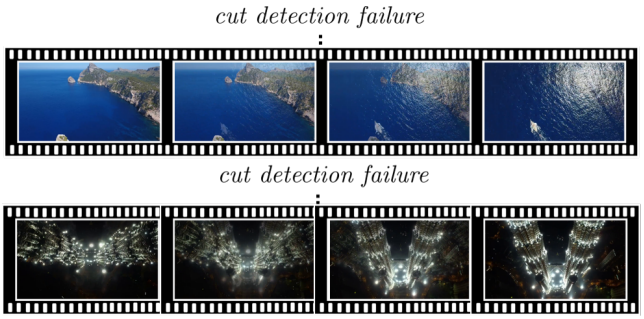


Figure 4: *Representative failure case of cut detection when two adjacent scenes are in similar color space.*

Failure cases: Representative examples of mis-cut detections are shown in Figure 4. Our failure cases suggest that if a cut happens between two sequential scenes which share similar color space, PySceneDetect, ffprobe, and ORB-SLAM2 might not detect the scene change. Hecate correctly detected some of these cuts between consecutive frames with a similar color-space. However, in Table 2, Hecate's precision is the lowest while it has the highest recall score among all other methods. This means Hecate reports many cuts with low precision and might detect all ground truth cuts as well as many wrong cuts. Table 2 also shows that the performance of the video cut detection methods are different from each other, and all the methods have some flaws in decomposing the edited drone videos into its clips. This can adversely impact the precision of filtered drone clips regardless of the cut detection method employed.

5. Training CNN on DVCD18K dataset

Our goal is to develop a content-aware drone video filtering algorithm for semantic search. To this end, leveraging our dataset annotations, we trained CNNs for five tasks (drone/non-drone, logo presence, time information, scene type, and location classification), evaluated their performance, and compared them to baseline methods. We use two types of CNN. The first type (named 2D CNN in the following) is "conventional" CNN based on 2D convolutional kernels to classify a single frame. The second type is (2+1)D CNN [TWT*18]: it accepts a set of frames as input and approximates 3D spatio-temporal convolutions by 2D and 1D convolutions. Overall, (2+1)D CNN is known to be more suitable for spatiotemporal feature learning compared to 2D CNN [TWT*18].

Data preparation: Among the 991 source videos, we randomly allocated 693 videos for training, 149 videos for validation, and 149 videos for testing. The duration of the training, validation, and testing sets was 31.27 hours (70.7% of the whole dataset), 6.94 hours (15.7%), and 6.04 hours (13.6%), respectively. We decompose the videos into clips according to the annotated cut times. Because time is written in unit of second during annotation, the first and last one second may contain frames from another clip. Therefore, we exclude them from each clip. To have balanced class distribution at the clip level, we discard videos of the dominant class to make the training, validation, and testing sets have the same number of clips belonging to each class.

CNN details: We use the popular CNN architecture DenseNet121 [HLvdMW17] pre-trained on ImageNet for 2D CNN and the R(2+1)D [TWT*18] architecture pre-trained on Kinetics 400 [KCS*17] for (2+1)D CNN. The input of the 2D CNN is downsized to the resolution of 144x192. For the (2+1)D CNN, the input is 16 consecutive frames, downsized to 112x112. One fully connected layer is added to the last convolution layer. We use sigmoid for binary classification and softmax for multi-class classification. All the networks are trained using the Adam optimizer [KB14], cross entropy loss, with a batch size of 10, for 5 epochs. We set the learning rate to 10^{-5} for 2D CNN and 10^{-6} for (2+1)D CNN.

Baseline methods: For baseline comparison, we used SVM with nonlinear kernels on HOG features [FP12]. To compute HOG, we use 10×10 window size, 10×10 block size, and 5×5 block stride on 100×100 resolution images. In addition to SVM, for indoor/outdoor classification, we also use the CNN-based scene recognition approach, Places365-CNN [ZLK*17] that returns a binary indoor/outdoor classification. We used the same training set to train SVM and our CNN, and used the same test set to evaluate SVM, Places365-CNN, and our CNNs. We did not intentionally re-train Places365-CNN to evaluate its performance on unseen drone shots and see how much it is generalizable for drone shots.

Evaluation results: The accuracy of each classification method is reported in Table 3. Additional evaluations, including precision, recall, and F-Score of our classifiers are available in Appendix 9.2. We compared 2D CNN, (2+1)D CNN, and SVM classifiers for all five tasks. Overall, CNNs outperform SVM by a substantial margin. We additionally compared our indoor/outdoor CNN classifier to the state-of-the-art Places365-CNN [ZLK*17]. The result shows that CNNs outperform Places365-CNN, even though Places365-CNN is based on a deeper ResNet152 structure [HZRS16] and trained on a larger dataset (1.8 million images) than ours (27,006 for training and 6,483 for validation). We believe that the underlying reason for the better performance of our networks is that they are trained on drone videos while the original training set of Places365-CNN does not contain any drone shots. This additionally motivates the need for our drone dataset to gain the better performance in automatic drone video classification.

In addition to the accuracy results, we computed and plotted the ROC curves (Figure 5). Time information and scene type classifiers have three categories. Therefore, they have three ROC curves by considering each class as positive. For all five tasks, (2+1)D CNN is the most outperforming method followed by 2D CNN and SVM. This can be explained by the fact that (2+1)D CNN is able to leverage temporal information from multiple input frames while 2D CNN and SVM operate on single frames.

Table 3: *Classification accuracy results.*

Type	Method	Accuracy
Clip category: drone/non-drone	SVM	0.68
	2D CNN	0.85
	(2+1)D CNN	**0.86**
Logo: yes/no	SVM	0.50
	2D CNN	0.71
	(2+1)D CNN	**0.81**
Time: day/night/between	SVM	0.52
	2D CNN	0.80
	(2+1)D CNN	**0.81**
Scene type: urban/rural/mixed	SVM	0.44
	2D CNN	0.61
	(2+1)D CNN	**0.64**
Location: indoor/outdoor	SVM	0.82
	2D CNN	0.88
	(2+1)D CNN	**0.89**
	Places365-CNN	0.84

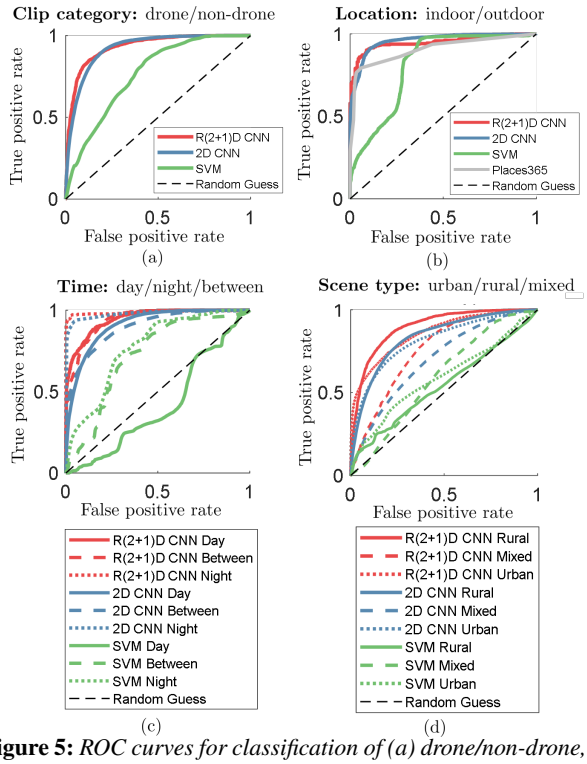
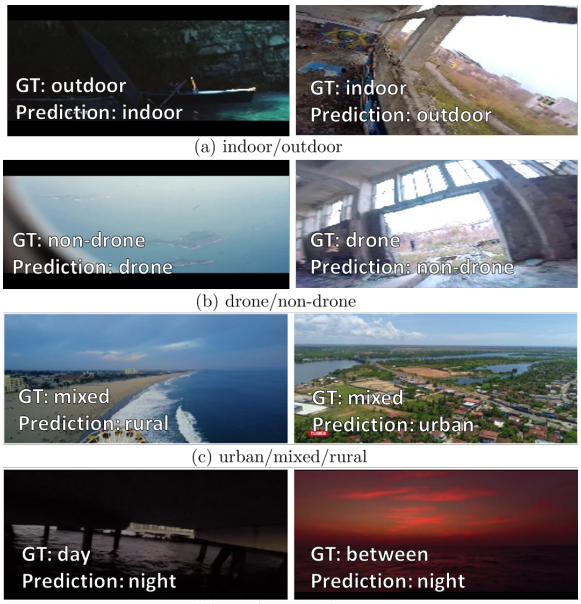


Figure 5: *ROC curves for classification of (a) drone/non-drone, (b) indoor/outdoor, (c) time information, and (d) scene type. The black diagonal line corresponds to random classification.*

Failure cases: Representative mis-classification results by our (2+1)D CNNs are shown in Figure 6. Images in (a) are mistakenly predicted as indoors and outdoors, respectively. The error in the left image might be due to the place where the video was captured. Specifically, the inside of a cave looks like an indoor environment. In the right image, the large area of outdoor environment captured by an indoor drone might be interpreted as an outdoor scene. Images in (b) are misclassified as drone and non-drone: the large pres-



(a) indoor/outdoor

(b) drone/non-drone

(c) urban/mixed/rural

(d) day/between/night

Figure 6: *Representative mis-classification results.*

ence of sky in the left image captured by an airplane passenger and the low altitude of the drone in the right image captured in a building might confuse the network. The left image in (c) belongs to the mixed (between urban and rural) category. However, this is classified as a rural place due to the small fraction of the buildings in the image. Similarly, the right image in (c) is mistakenly classified as an urban area while it belongs to the mixed category. For the time classifier, when a shot is captured during the day from a dark place with low exposure to light, network might mistakenly classify the time as night. In addition, sometimes during a sunset or sunrise, the sky might look brighter (or darker) if the sunrise or sunset are about to be started (or finished). This can also cause some mistakes in our time classifier, as shown in Figure 6(d).

6. Automatic drone video collection filtering

As briefly mentioned in Section 3, a high percentage of Youtube videos related to drones is composed of non-drone video clips. As a quantitative evidence, we manually measured that 86.9% of the duration of the top 100 Youtube videos with the search query “flying drone" is composed of non-drone video clips, for example people talking about drones in the video. To overcome this challenge and facilitate the search of drone video clips, we present an approach which operates on an unstructured collection of videos, for example Youtube videos with the search query “flying drone", and automatically returns clips captured by drones. This requires to automatically decompose the videos into individual clips and identify the drone clips. A representative example of Youtube video frames and drone clips filtered by our algorithm is available in Figure 7. Our algorithm is tuned to aggressively prune the input video collection and conservatively extracts only the true drone shots. This is conducted at the expense of leaving out some clips captured by drones but for which our prediction has not gained sufficient confidence. This process is sometimes called dataset distillation [AEWQ*15], and it is motivated by the use case where a user searches for *some* correct videos instead of *all* potentially corrupted videos.

In addition to drone clip retrieval, our approach enables users to search for drone video clips with high-level semantic search queries, such as “outdoor drone clips captured from rural places during the day". To this end, we employ our trained CNN to estimate the captured time, location, and scene-type of drone videos. See the dedicated labels for each drone clip in Figure 7.

We trained and developed our drone video filtering algorithm using AIRVŪZ data. However, to make our experiment similar to the real world production scenario case, we tested the validity of our method on unseen YouTube videos and verified our filtering method can generalize well on filtering unseen videos. To evaluate our video filtering algorithm, we followed the standard evaluations typically used for a distillation algorithm [AEWQ*15] and conducted five evaluations in Section 6.3.

6.1. Proposed approach

We now present our approach to retrieve drone video clips among a video collection. In our initial experiments, we first decomposed the video collection into individual clips via four out-of-the-box cut detection methods (Section 4) and then identified the drone clips by our drone/non-drone classification CNN (Section 5). The initial results showed that the precision of this approach can be low or different depending on various cut detection methods employed. See Section 6.3 for more details. The underlying reason is that a video cut method might mistakenly group drone/non-drone clips together. For example, such a case may happen if two adjacent clips are in similar colour space.

To make our algorithm robust and independent of defects in cut detection methods, we propose a “reverse" approach in which we first identify drone frames (frame-level drone classification) and then pass them to a drone content-aware cut detection method (drone frame block detection). The detail of these two steps are as follows:

1) We classify each frame from the video collection using our drone/non-drone CNN classifier (Section 5). Our algorithm classifies each frame based on its predicted probability of being a drone frame, reported by the last layer of our CNN drone classifier. When this probability is higher than a threshold (e.g., 0.8), we decide it as a drone frame. We conservatively tune this threshold to return the true drone frames for which our CNN gained sufficient confidence.
2) We automatically detect series of consecutive drone frames and consider them as a candidate set of drone video clips, as shown in Figure 8. For each candidate drone video clip, we also compute the mean and median values of the probabilities (of being captured by a drone) of its frames. If both the mean and median values are higher than a threshold (e.g., 0.9), we add the clip to the final set of drone video clips. These two steps extract the true drone video clips from a video collection in the wild.

For high-level semantic search, we conduct the following additional step:

3) For each identified drone clip, we employ our CNN (Section 5) to predict additional labels such as time information, scene type category, logo presence, and shot location. See Evaluation 3 in Section 6.3 for more details.

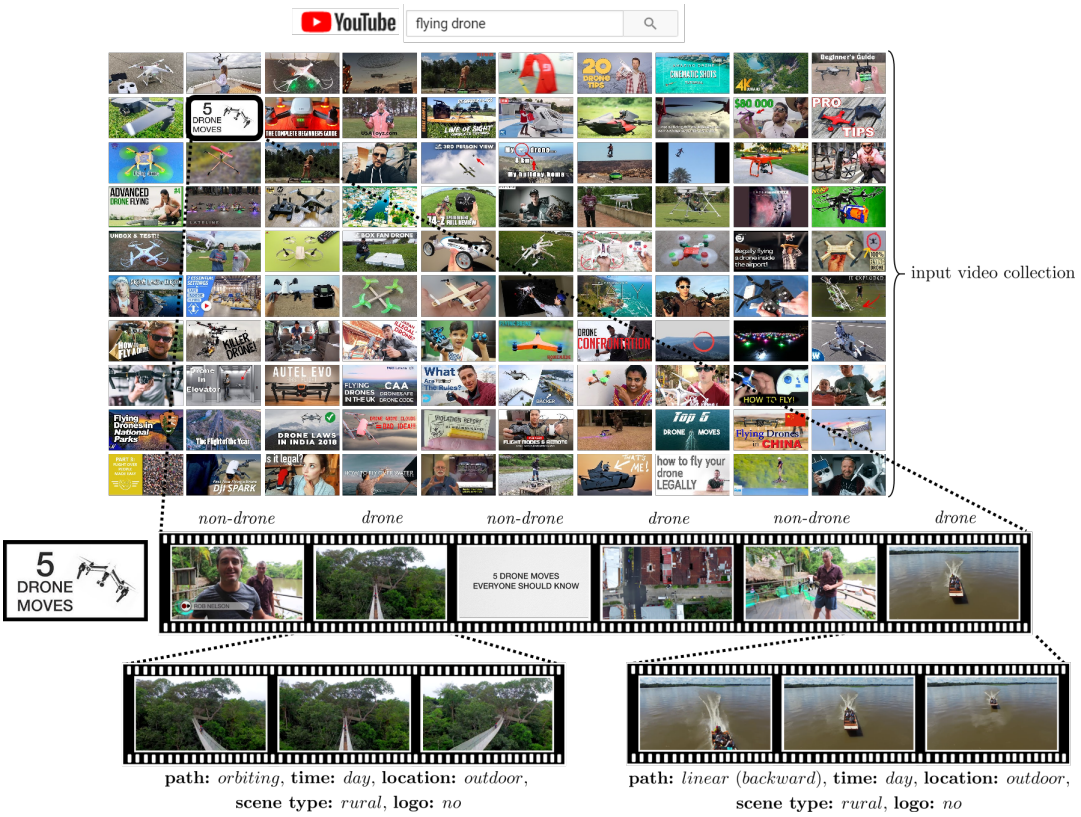


Figure 7: *Representative example of Youtube video collection filtering. Given an input video collection, we automatically decompose the videos into clips and classify them into drone vs. non-drone clips by our trained CNN. For each detected drone clip, we predict additional labels such as time information, scene type category, existence of a logo, path category, and shot location (indoor/outdoor). This approach can be used to automatically filter and retrieve specific types of drone video clips from an unlabeled video collection using high-level search queries, such as "drone video shot outdoor in day time with an orbiting motion".*

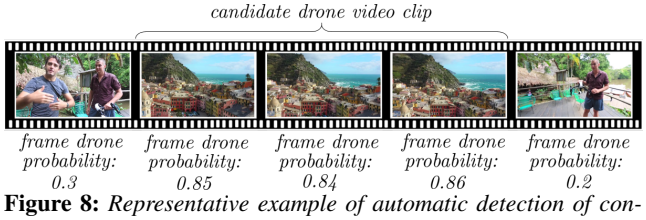


Figure 8: *Representative example of automatic detection of consecutive drone frames with high drone probability as a candidate drone video clip.*

4) For each identified drone clip, we also compute the drone trajectory using Metashape (Section 3.3). Moreover, we identify some types of drone trajectories, for example the orbiting or linear (forward/backward) path shapes. See Evaluation 5 in Section 6.3 for more details.

6.2. Evaluation procedure

We now conduct quantitative and qualitative evaluations of our distillation algorithm. The quantitative evaluation compared our estimated CNN results with ground truth drone clips. For the evaluations, we downloaded the first top 100 videos from Youtube with the search query "flying drone", as shown in Figure 7. We ran our approach to automatically identify the drone clips and computed the labels such as scene type and time information. To obtain ground truth, we asked annotators to manually extract the drone clips by defining cut times, and annotate them by logo/non-logo, scene type, time information, and location labels. Finally, we computed the precision of our automatic approach by comparing our CNN network results with the ground truth. To compute the precision of our algorithm, we divided the number of "correctly predicted" drone clips by the total number of predictions. A clip is considered "correctly predicted" when its Intersection over Union (IoU) metric with respect to the ground truth drone clip is higher than a threshold (e.g., 0.95). Intersection over Union (IoU) metric is widely used [RTG*19] as a basis to evaluate segmentation [AMM*18, COR*16, LMB*14, ZZP*17], object detection [EVGW*10, LMB*14], and tracking [LTMR*15] tasks. To obtain IoU metric, we compute the intersection between the predicted and ground truth drone clips (the duration of overlap between them) divided by their union (the duration encompassed by both of them).

6.3. Evaluation results

The precision of our method is shown in Figure 9. It shows that when we put a conservative threshold to detect drone frames (e.g., frames with a drone probability higher than 0.95), the precision of the filtered drone clips is 1.

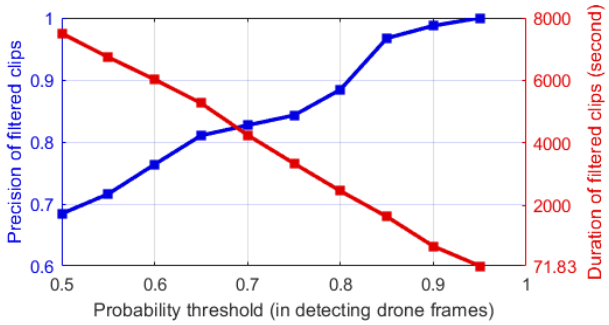


Figure 9: *Evaluating the trade-off between the distilled set precision and its duration.*

[Evaluation 1] Size vs. quality of the filtered drone clips: To demonstrate the trade-off between the distilled set size and the precision score of the obtained filtered drone clips, we compared the ground truth annotations with our drone/non-drone CNN classifier results. In our CNN classifier, if the probability of being a drone frame is higher than a threshold (conservativeness metric), we consider it as a drone frame. Using a higher value of the threshold results in a more conservative video filtering in which our network gained higher confidence in detecting true drone frames. We examined the conservativeness of our algorithm by considering different probability threshold values in the range [0.5-1]. Figure 9 plots the trade-off between size and precision. It shows that a more conservative setting allows for a nearly perfect, but smaller, distilled drone video clip collection. This trade-off based on the conservativeness metric provides a systematic way to tune the duration and precision of filtered clips. In addition, Figure 9 confirms that when tuned conservatively, our automatic filtering algorithm is able to extract a near perfect subset of true drone videos.

[Evaluation 2] Scaling to large distilled drone video clips: We designed experiments to analyze the scaling behavior of our filtering algorithm. To this end, we examined the precision and duration of filtered videos as a function of the input video collection duration. Starting with the full size of our 100 Youtube videos, we generated different input video collection durations by randomly selecting a set of videos. For each set of videos, we ran our algorithm and measured the precision and duration of filtered videos. Figure 10 shows that the duration of the filtered drone clips is approximately proportional to the duration of the input videos (left), while the precision score remains nearly constant (right). This suggests that a larger drone video clip set can be obtained, from a larger input video collection size while the precision of the returned drone clips remains similar. Because we randomly select sets of videos for this experiment, in Figure 10, we show median, 0.25, and 0.75 quantile of the duration (left) and precision (right) of the filtered clips, calculated over various input video sets.

[Evaluation 3] Semantic search: We developed an automatic approach that enables a user to narrow down the search results with high-level semantic search queries. To this end, first, we apply a simple word segmentation of the query sentence. Then, among the segmented words, we find words that correspond to the classes/keywords of our classifiers (e.g., indoor/outdoor). Finally, we search for consecutive drone frames for which our CNN predictors gain sufficient confidence in them to correspond to detected classes/keywords. For example, to extract "outdoor drone clips"

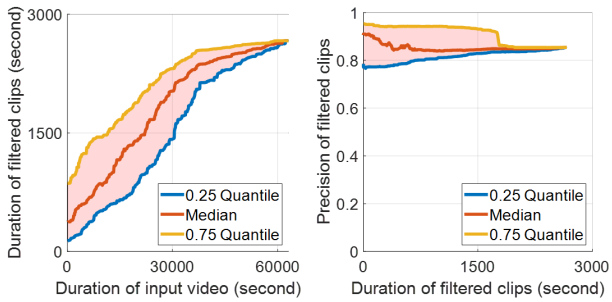


Figure 10: *Scaling evaluation. Duration (left) and precision (right) of filtered drone clips by increasing the input video collection duration.*

(detected classes/keywords: "outdoor" and "drone"), we search for consecutive frames with high probability of being the both "outdoor" and "drone" frames in our CNN predictors. To evaluate the precision of our semantic search results, we ran our approach to extract (a) "outdoor drone clips", (b) "drone clips captured during the sunset/sunrise (between)", (c) "urban drone clips", (d) "non-logo drone clips", (e) "outdoor non-logo drone clips captured from urban places during the day" and (f) "outdoor drone clips captured from a mixed urban and rural place during the day", as shown in Figure 12. We gained perfect precision of 1 by conservatively tuning our approach.

The precision of our semantic drone video search is shown in Figure 11. It confirms that our distillation approach gains perfect precision of 1, if we tune it conservatively to return video clips for which our CNN predictors gained high confidence in them to correspond to the search queries (a) to (f). Moreover, our distilled video results with search queries "outdoor drone video shot with (a) orbiting motion, (b) linear (forward) and linear (backward)" are shown in Figure 14. We gained perfect precision of 1 by conservatively tuning our approach.

[Evaluation 4] Automatic drone video collection filtering via initial cut detection: As mentioned in Section 6.1, we present a distillation method which automatically filters video collections to return true drone shots with its associated information such as time, location, and scene type, as shown in Figure 7. In our early experiments, we first decomposed the video collection into clips via a cut detection method (Section 4). We used PySceneDetect, ffprobe, Hecate, and ORB-SLAM2 to detect the cuts. Then, we lever-

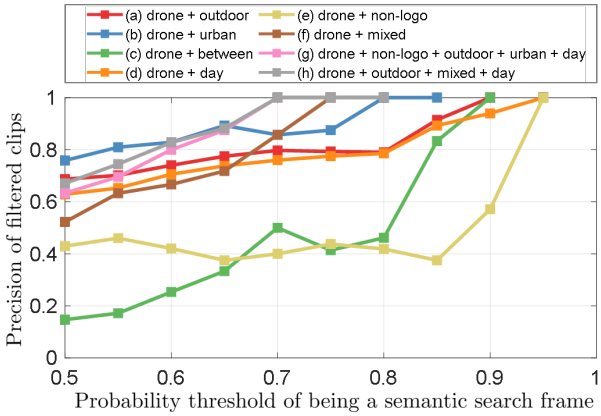


Figure 11: *Precision of semantic search results.*

Figure 12: *Representative semantic search results with the search queries (a) to (f).*

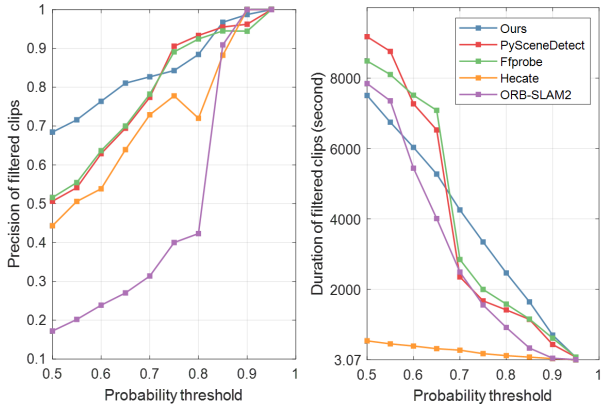


Figure 13: *Precision (left) and duration (right) of filtered drone clips associated with different methods employed.*

aged our drone/non-drone CNN (Section 5) to detect true drone clips among identified clips. For each identified clip, we compute the mean probability of its frames to be a drone frame, and if it is higher than a threshold, we return it as a distilled drone clip. Figure 13 shows the precision of the distillation algorithm using different cut detection methods. It confirms that the precision and duration of returned distilled drone clips can differ and even be low depending on various cut detection methods employed. To address this problem and make our distillation approach robust to initial cut detection methods, we return consequent frames with high probability of being drone frames as a distilled drone clip , as shown in Figure 8. In this case, probability threshold defines whether each frame can be considered as a drone frame, and then we search for consequent drone frames. As shown by the blue line in Figure 13, our method leads to a higher precision in the majority of probability thresholds. It also returns longer duration of distilled drone clips with high precision. See the precision and duration of filtered drone clips at probability thresholds higher than 0.8 in Figure 13.

[Evaluation 5] Drone path shape: In the context of Youtube video filtering based on the drone camera path shape, we identify some types of drone trajectories, for example the orbiting or linear (forward/backward) path shapes. To identify these path shapes, we use RANSAC to fit a circle and a line for each drone camera path computed by Metashape. By comparing the percentage of RANSAC inliers[¶] for the best fitted circle and line, we define whether the camera path is more similar to the circular or linear path. If the percentage of inliers for both fitted circle and line are less than a threshold (80%), we define it as a non-categorized path. We also put a threshold on the circular path coverage in such a way that the circular path arc should include at least 50% of the fitted circle. Similarly, the linear path length is handled by enforcing that the linear path length must be higher than a threshold. For the linear paths, we also compute the moving direction to determine if the camera moved forward or backward. To this end, we compute the angle between the camera view vector and the vector from the initial point to the end point on the path for each path point. Then, if the majority of these angles is lower (respectively higher) than 90°, we consider the camera direction as forward (respectively backward). Our distilled video results based on the drone camera path shape and with search queries "outdoor drone video shot with (a) orbiting motion, (b) linear (forward) and linear (backward)" are shown in Figure 14. We gained perfect precision of 1 by conservatively tuning our approach.

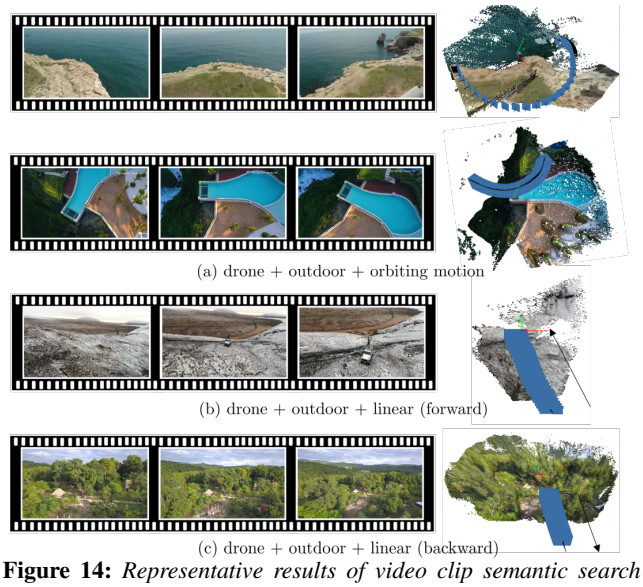


Figure 14: *Representative results of video clip semantic search among Youtube videos. For each result, we indicate the semantic search keywords and show some frames of the filtered clips as well as the 3D reconstruction and drone path.*

[¶] RANSAC inliers are path points that fit the RANSAC model, i.e., distances between the points and fitted circle (or line) are less than a threshold. For a circular path, we define this threshold based on the fitted circle radius (e.g., 1% of the fitted circle radius), while we use a constant threshold for a linear path.

7. Future work

We used our dataset for video cut detection (Section 4), classification of drone vs. non-drone, scene type (rural/mixed/urban), time information (day/between/night), location (indoor/outdoor), and logo vs. non-logo presence in drone videos (Section 5), and semantic drone video search (Section 6). Because of the provided annotations in our dataset, one can leverage our dataset for the following tasks:

Drone path planning: Our dataset contains drone video clips with their corresponding drone camera paths. One can leverage this data to train deep learning models to generate drone paths, given drone camera views. In addition, because our dataset contains various drone shots, one can analyze drone camera paths to detect global patterns and shot styles used in drone cinematography. Our compilation of the largest dataset provides drone camera paths of *edited* videos, captured by human camera operators and *edited* to contain aesthetically pleasing drone shots and camera motions.

Popularity prediction of drone videos: The popularity of a drone video is affected by many factors, such as aesthetical aspects, content type, camera motion pattern, and video storytelling. While several methods have been presented for aesthetic assessment and popularity prediction [HGS19,THP*17], they are designed for general image/video content. Because our dataset contains social platform metadata such as number of likes and views for each video, an interesting direction for future work is to investigate new deep learning based approaches specifically dedicated to automatic popularity prediction of *drone* shots.

Music recommendation for drone videos: In contrast to other datasets designed for specific tasks [KBKE18,BMS*17,DQY*18], our dataset contains diverse high-quality professional *edited* drone videos. These videos have both visual and audio data in which the overlaid music is selected by the video editor, for example to accompany the video storytelling and dramatize the video content. An interesting direction is to leverage the audio-visual data, for example to automatically suggest a musical piece for a given drone shot.

Recommendation of video transition effects: Because we annotate video transition effects between consecutive video clips, one can use our dataset to train deep learning models to recommend a video transition effect, commonly preferred by video editors considering content of consecutive video clips.

Improving drone video classification: In the context of drone video analysis, we used our CNN to classify drone clips based on drone/non-drone, indoor/outdoor, scene type, and time information. Our CNN results show that there is room for classification improvement. An interesting approach is to explicitly include motion information to the network, for example in the form of optical flow, in addition to the visual content. To improve the generalization of CNN classifiers outside the training set, Peterson et al. [PBGR19] proposed to use the human uncertainty in annotating the data. A direction worthwhile to explore is to use soft labels provided through human confusion as a replacement for one-hot label encodings. For example, in our case, first, different annotators sometimes had different perception for scene type (urban/rural/mixed) class of a given frame. Second, in our dataset, we provide a confidence score for clip categories (drone/non-drone) as an indicator of the annotators confidence in the validity of labels (Section 3.1). One could use these perceptual uncertainties to train with full label distributions and make more robust classifiers.

8. Conclusion

The first step toward learning automated drone cinematography from videos in the wild is to filter drone clips and extract their camera motions out of these videos that commonly contain both drone and non-drone content. In this paper, we proposed the first method that can automatically retrieve drone clips from an unlabeled video collection using high-level search query, such as “drone video shot outdoor in day time". The retrieved clips also contain camera motions, camera view, and 3D reconstruction of a scene that can help develop intelligent camera path planners. We conducted five qualitative and quantitative evaluations and showed the validity of our drone video distillation algorithm. We showed that our method has a mechanism to only return true drone clips with perfect precision of 1, and can be scaled to create large distilled drone video clip datasets with high precision. We examined the validity of our extracted drone path shapes by successfully filtering drone clips with specific drone camera motions.

In addition, we have proposed DVCD18K, a novel, large-scale, annotated dataset composed of more than 18,000 drone video clips, which leads to a total amount of more than 44 hours of edited drone videos. Our annotations include various levels of information such as clip category (drone/non-drone), scene description, location, video editing information, social platform metadata, and 3D camera path among many others. The compiled clips offer a great variety in terms of location, appearance, drone camera motion, scene category, shot type, quality, and editing effect. Our dataset is the first large-scale, fully annotated dataset of *edited* drone videos that contains aesthetically pleasing camera motions and views to develop intelligent drone path planners. We hope that our dataset will contribute to the development of algorithms for automated drone cinematography and drone video analysis.

Acknowledgements

This research was supported by Culture, Sports and Tourism R&D Program through the Korea Creative Content Agency grant funded by the Ministry of Culture, Sports and Tourism in 2022 (Project Name: Development of self-evolving AI Creation Platform, Project Number: R2020040180, Contribution Rate:100%). We specially thank Dr. Jean-Charles Bazin for his valuable insights.

References

[AEWQ*15] AVERBUCH-ELOR H., WANG Y., QIAN Y., GONG M., KOPF J., ZHANG H., COHEN-OR D.: Distilled collections from textual image queries. In *CGF* (2015). 8

[Agi] AGISOFT METASHAPE:. URL: www.agisoft.com. 5

[AMM*18] ALHAIJA H. A., MUSTIKOVELA S. K., MESCHEDER L., GEIGER A., ROTHER C.: Augmented reality meets computer vision: Efficient data generation for urban driving scenes. *IJCV 126*, 9 (2018), 961–972. 9

[ASN*20] ASHTARI A., STEVŠIĆ S., NÄGELI T., BAZIN J.-C., HILLIGES O.: Capturing subjective first-person view shots with drones for automated cinematography. *TOG 39*, 5 (2020), 1–14. 1

[BBS*21] BONATTI R., BUCKER A., SCHERER S., MUKADAM M., HODGINS J.: Batteries, camera, action! learning a semantic control space for expressive robot cinematography. In *ICRA* (2021), IEEE, pp. 7302–7308. 1

[BKRE15] BONETTO M., KORSHUNOV P., RAMPONI G., EBRAHIMI T.: Privacy in mini-drone based video surveillance. In *ICIP* (2015). 3

[BMS*17] BAREKATAIN M., MARTÍ M., SHIH H., MURRAY S., NAKAYAMA K., MATSUO Y., PRENDINGER H.: Okutama-Action: An aerial view video dataset for concurrent human action detection. In *CVPR* (2017). 3, 12

[Cas18] CASTELLANO B.: PySceneDetect. http://py.scenedetect.com. 6

[CKL*18] CHEN C., KUO Y., LEE T., LEE C., HSU W. H.: Drone-view building identification by cross-view visual learning and relative spatial estimation. In *CVPR Workshop* (2018), pp. 1477–1485. 1

[COR*16] CORDTS M., OMRAN M., RAMOS S., REHFELD T., ENZWEILER M., BENENSON R., FRANKE U., ROTH S., SCHIELE B.: The cityscapes dataset for semantic urban scene understanding. In *CVPR* (2016), pp. 3213–3223. 9

[ÇTM18] ÇIGLA C., THAKKER R., MATTHIES L. H.: Onboard stereo vision for drone pursuit or sense and avoid. In *CVPR Workshop* (2018), pp. 625–633. 1

[DQY*18] DU D., QI Y., YU H., YANG Y., DUAN K., LI G., ZHANG W., HUANG Q., TIAN Q.: The unmanned aerial vehicle benchmark: Object detection and tracking. In *ECCV* (2018). 3, 12

[EVGW*10] EVERINGHAM M., VAN GOOL L., WILLIAMS C. K., WINN J., ZISSERMAN A.: The Pascal Visual Object Classes (VOC) Challenge. *IJCV 88*, 2 (2010), 303–338. 9

[FFm18] FFMPEG DEVELOPERS: FFmpeg tool. http://ffmpeg.org/. 6

[FP12] FORSYTH D. A., PONCE J.: *Computer Vision - A Modern Approach, Second Edition*. Pitman, 2012. URL: http://vig.pearsoned.com/store/product/1,1207,store-12521_isbn-013608592X,00.html. 7

[GFTG16] GALVANE Q., FLEUREAU J., TARIOLLE F., GUILLOTEL P.: Automated cinematography with unmanned aerial vehicles. In *WICED* (2016), pp. 23–30. 1

[GH18] GEBHARDT C., HILLIGES O.: WYFIWYG: Investigating Effective User Support in Aerial Videography. In *arXiv preprint arXiv:1801.05972* (2018). 1

[GH21] GEBHARDT C., HILLIGES O.: Optimization-based user support for cinematographic quadrotor camera target framing. In *ACM CHI* (2021), pp. 1–13. 1

[GHN*16] GEBHARDT C., HEPP B., NÄGELI T., STEVŠIĆ S., HILLIGES O.: Airways: Optimization-based planning of quadrotor trajectories according to high-level user goals. In *ACM CHI* (2016), pp. 2508–2519. 1

[GLC*18] GALVANE Q., LINO C., CHRISTIE M., FLEUREAU J., SERVANT F., TARIOLLE F., GUILLOTEL P., ET AL.: Directing cinematographic drones. *TOG 37*, 3 (2018), 34. 1

[GMV*18] GEBRU T., MORGENSTERN J., VECCHIONE B., VAUGHAN J. W., WALLACH H., DAUMÉ III H., CRAWFORD K.: Datasheets for datasets. *arXiv preprint arXiv:1803.09010* (2018). 4

[GSH18] GEBHARDT C., STEVSIC S., HILLIGES O.: Optimizing for aesthetically pleasing quadrotor camera motion. *TOG 37*, 4 (2018), 90:1–90:11. 1

[HGS19] HOSU V., GOLDLUCKE B., SAUPE D.: Effective aesthetics prediction with multi-level spatially pooled features. In *CVPR* (2019), pp. 9375–9383. 12

[HLH17] HSIEH M., LIN Y., HSU W. H.: Drone-based object counting by spatially regularized regional proposal network. In *ICCV* (2017), pp. 4165–4173. 1, 3

[HLvdMW17] HUANG G., LIU Z., VAN DER MAATEN L., WEINBERGER K. Q.: Densely connected convolutional networks. In *CVPR* (2017). 7

[HLY*19] HUANG C., LIN C.-E., YANG Z., KONG Y., CHEN P., YANG X., CHENG K.-T.: Learning to film from professional human motion videos. In *CVPR* (2019). 1, 3

[HZRS16] HE K., ZHANG X., REN S., SUN J.: Deep residual learning for image recognition. In *CVPR* (2016). 7

[JCW*21] JIANG H., CHRISTIE M., WANG X., LIU L., WANG B., CHEN B.: Camera keyframing with style and control. *TOG 40*, 6 (2021), 1–13. 1

[JJG*16] JOUBERT N., JANE L. E., GOLDMAN D. B., BERTHOUZOZ F., ROBERTS M., LANDAY J. A., HANRAHAN P.: Towards a drone cinematographer: Guiding quadrotor cameras using visual composition principles. *arXiv preprint arXiv:1610.01691* (2016). 1

[JRT*15] JOUBERT N., ROBERTS M., TRUONG A., BERTHOUZOZ F., HANRAHAN P.: An interactive tool for designing quadrotor camera shots. *TOG 34*, 6 (2015), 238:1–238:11. 1

[JWW*20] JIANG H., WANG B., WANG X., CHRISTIE M., CHEN B.: Example-driven virtual cinematography by learning camera behaviors. *TOG 39*, 4 (2020), 45–1. 1

[KB14] KINGMA D. P., BA J.: Adam: A method for stochastic optimization. *ICLR* (2014). URL: https://arxiv.org/abs/1412.6980. 7

[KBKE18] KRAJEWSKI R., BOCK J., KLOEKER L., ECKSTEIN L.: The highD dataset: A drone dataset of naturalistic vehicle trajectories on German highways for validation of highly automated driving systems. In *ITSC* (2018). 3, 12

[KCS*17] KAY W., CARREIRA J., SIMONYAN K., ZHANG B., HILLIER C., VIJAYANARASIMHAN S., VIOLA F., GREEN T., BACK T., NATSEV P., SULEYMAN M., ZISSERMAN A.: The kinetics human action video dataset. *arXiv preprint arXiv:1705.06950 abs/1705.06950* (2017). URL: http://arxiv.org/abs/1705.06950, arXiv:1705.06950. 7

[KP12] KRULIKOVSKÁ L., POLEC J.: An efficient method of shot cut detection. *Int. J. of Computer and Information Engineering 6*, 3 (2012), 63–79. 6

[KSI*19] KALJAHI M. A., SHIVAKUMARA P., IDRIS M. Y. I., ANISI M. H., LU T., BLUMENSTEIN M., NOOR N. M.: An automatic zone detection system for safe landing of UAVs. *Expert Syst. Appl.* (2019). 1

[LMB*14] LIN T.-Y., MAIRE M., BELONGIE S., HAYS J., PERONA P., RAMANAN D., DOLLÁR P., ZITNICK C. L.: Microsoft COCO: Common objects in context. In *ECCV* (2014), pp. 740–755. 9

[LTMR*15] LEAL-TAIXÉ L., MILAN A., REID I., ROTH S., SCHINDLER K.: MOTChallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942* (2015). 9

[MSG16] MUELLER M., SMITH N., GHANEM B.: A benchmark and simulator for UAV tracking. In *ECCV* (2016). 3

[MT17] MUR-ARTAL R., TARDÓS J. D.: ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras. *TRO* (2017). 5, 6

[NAMD*17] NÄGELI T., ALONSO-MORA J., DOMAHIDI A., RUS D., HILLIGES O.: Real-time motion planning for aerial videography with dynamic obstacle avoidance and viewpoint optimization. *RAL 2*, 3 (2017), 1696–1703. 1

[NMD*17] NÄGELI T., MEIER L., DOMAHIDI A., ALONSO-MORA J., HILLIGES O.: Real-time planning for automated multi-view drone cinematography. *TOG 36*, 4 (2017), 132:1–132:10. 1

[NMRB17] NIKOLAIDIS N., MADEMLIS I., RAPTOPOULOU C., BULL D.: Tutorial on drone vision for cinematography. In *ICCV Tutorial* (2017). 1

[NOP*18] NÄGELI T., OBERHOLZER S., PLÜSS S., ALONSO-MORA J., HILLIGES O.: Flycon: real-time environment-independent multi-view human pose estimation with aerial vehicles. *TOG 37*, 6 (2018), 182:1–182:14. 1

[PBGR19] PETERSON J. C., BATTLEDAY R. M., GRIFFITHS T. L., RUSSAKOVSKY O.: Human uncertainty makes classification more robust. In *ICCV* (2019), pp. 9617–9626. 12

[Res15] RESTAS A.: Drone applications for supporting disaster management. *WJET* (2015). 1

[RH16] ROBERTS M., HANRAHAN P.: Generating dynamically feasible trajectories for quadrotor cameras. *TOG 35*, 4 (2016), 61:1–61:11. 1

[RSAS16] ROBICQUET A., SADEGHIAN A., ALAHI A., SAVARESE S.: Learning social etiquette: Human trajectory understanding in crowded scenes. In *ECCV* (2016). 3

[RSD*17] ROBERTS M., SHAH S., DEY D., TRUONG A., SINHA S. N., KAPOOR A., HANRAHAN P., JOSHI N.: Submodular trajectory optimization for aerial 3D scanning. In *ICCV* (2017), pp. 5334–5343. 1

[RTG*19] REZATOFIGHI H., TSOI N., GWAK J., SADEGHIAN A., REID I., SAVARESE S.: Generalized intersection over union: A metric and a loss for bounding box regression. In *CVPR* (2019), pp. 658–666. 9

[SMGH18] SMITH N., MOEHRLE N., GOESELE M., HEIDRICH W.: Aerial path planning for urban scene reconstruction: a continuous optimization method and benchmark. *TOG 37*, 6 (2018), 183:1–183:15. 1

[SPO18] SINGH A., PATIL D., OMKAR S. N.: Eye in the sky: Real-time drone surveillance system (DSS) for violent individuals identification using scatternet hybrid deep learning network. In *CVPR Workshop* (2018), pp. 1629–1637. 1

[SRVJ16] SONG Y., REDI M., VALLMITJANA J., JAIMES A.: To click or not to click: Automatic selection of beautiful thumbnails from videos. *arXiv preprint arXiv:1609.01388* (2016). 6

[SVB*20] SCHEIRER W., VIDALMATA R., BANERJEE S., RICHARD-WEBSTER B., ALBRIGHT M., DAVALOS P., MCCLOSKEY S., MILLER B., TAMBO A., GHOSH S., ET AL.: Bridging the gap between computational photography and visual recognition. *TPAMI* (2020). 3

[TBLA16] TRUONG A., BERTHOUZOZ F., LI W., AGRAWALA M.: Quickcut: An interactive tool for editing narrated video. In *UIST* (2016), pp. 497–507. 1

[TC14] TREMAYNE M., CLARK A.: New perspectives from the sky. *Digit. Journal.* (2014). 1

[THP*17] TANG L., HUANG Q., PUNTAMBEKAR A., VIGFUSSON Y., LLOYD W., LI K.: Popularity prediction of Facebook videos for higher quality streaming. In {*USENIX*} *Annual Technical Conference* (2017), pp. 111–123. 12

[TWT*18] TRAN D., WANG H., TORRESANI L., RAY J., LECUN Y., PALURI M.: A closer look at spatiotemporal convolutions for action recognition. In *CVPR* (2018). 6, 7, 14

[WYH*19] WANG M., YANG G.-W., HU S.-M., YAU S.-T., SHAMIR A., ET AL.: Write-a-video: computational video montage from themed text. *TOG 38*, 6 (2019), 177–1. 1

[XYH*18] XIE K., YANG H., HUANG S., LISCHINSKI D., CHRISTIE M., XU K., GONG M., COHEN-OR D., HUANG H.: Creating and chaining camera moves for quadrotor videography. *TOG 37*, 4 (2018), 88. 1

[ZLK*17] ZHOU B., LAPEDRIZA A., KHOSLA A., OLIVA A., TORRALBA A.: Places: A 10 million image database for scene recognition. *TPAMI* (2017). 7, 14

[ZLP*18] ZHOU X., LIU S., PAVLAKOS G., KUMAR V., DANIILIDIS K.: Human motion capture using a drone. In *ICRA* (2018), pp. 2027–2033. 1

[ZTF*18] ZHOU T., TUCKER R., FLYNN J., FYFFE G., SNAVELY N.: Stereo magnification: learning view synthesis using multiplane images. *TOG* (2018). 6

[ZWB*18] ZHU P., WEN L., BIAN X., HAIBIN L., HU Q.: Vision meets drones: A challenge. *arXiv preprint arXiv:1804.07437* (2018). 3

[ZZP*17] ZHOU B., ZHAO H., PUIG X., FIDLER S., BARRIUSO A., TORRALBA A.: Scene parsing through ADE20K dataset. In *CVPR* (2017), pp. 633–641. 9

9. Appendix

9.1. Path computation time

Our dataset contains time consuming automatic annotations such as the path computation and 3D reconstruction with Metashape and ORB-SLAM2. Figure 16 shows the automatic path computation time with (a) Metashape and (b) ORB-SLAM2. See Section 3.3 for more details. Figure 16-left plots the tendency between the duration of input video clips and path computation. It shows a highly linear tendency for ORB-SLAM2, and overall, the longer clip duration requires the longer path computation time. Figure 16-right demonstrates the approximately linear tendency between the cumulative duration of input video clips with respect to the cumulative path computation time. On average, it took 9.3 hours to annotate one hour of source video for automatic annotation. The automatic annotation of the whole dataset took 410 hours of computation (320 hours for Metashape and 90 hours for ORB-SLAM2). Figure 15 shows representative results of the path computation and 3D reconstruction with Metashape.

9.2. CNN supplementary

In Section 5, we only reported the accuracies of our classifiers. However, the classification accuracy alone is not sufficient to judge the effectiveness of a classifier. Here, we provide the precision, recall, and F-score for the classification of drone/non-drone, logo/non-logo, shot location, time information, and scene type by different approaches such as SVM, 2D CNN, (2+1)D CNN [TWT*18], and Places365-CNN [ZLK*17]. Because the scene type and time information have multiple classes and therefore are not subject to a binary classification, we separately compute the precision, recall, and F-score for each class. To this end, we consider one class (e.g., night) as a positive class and others (e.g., day and between) as a negative class. See the evaluation metrics of the night class in Table 4. For the final evaluation, we compute the mean value of the precision, recall, and F-score for every class. Specifically, as an example, the final F-score of the time information classifier using SVM is computed by averaging out the SVM day/between/night F-scores in Table 4. Finally, Table 5 shows the precision, recall, and F-score for all of our classifiers. F-score can provide an improved measure of the performance of a classifier by using both precision and recall. Based on the F-score in Table 5, for all five tasks, (2+1)D CNN is the most outperforming method followed by 2D CNN and SVM. This can be explained by the fact that (2+1)D CNN is able to leverage temporal information from multiple input frames while 2D CNN and SVM operate on single frames. When running our experiments on one to two GTX 1080Ti GPUs, the training time of each experiment took less than a day.

9.3. Social data

Figure 17 shows the social data, i.e., the number of likes (left) and comments (right). 80% of videos received less than 41 likes and 32 comments. 5% of videos received more than 66 likes and 52 comments. Mean, median, min, and max values for the number of likes and comments are (29, 25, 2, 229) and (22, 19, 0, 107), respectively. The top 10^{th} and 90^{th} percentiles are (52, 10) and (43, 6) for likes and comments, respectively.

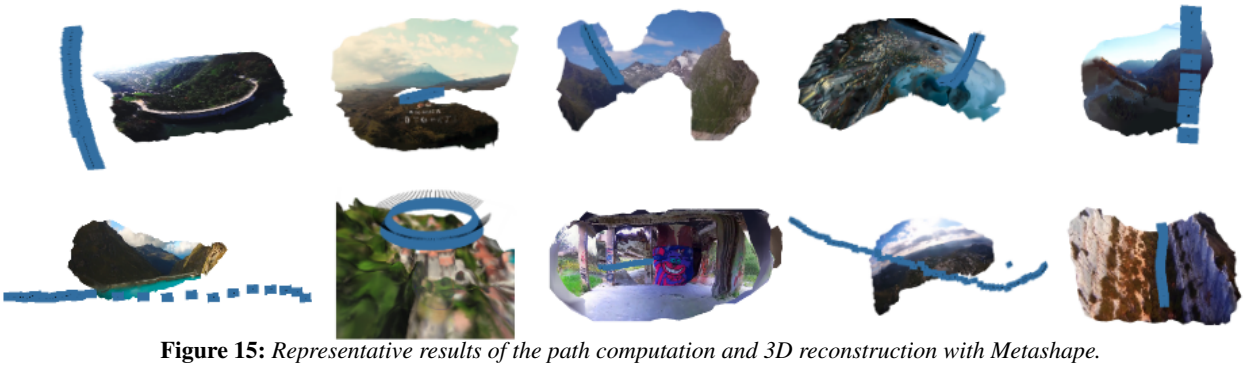
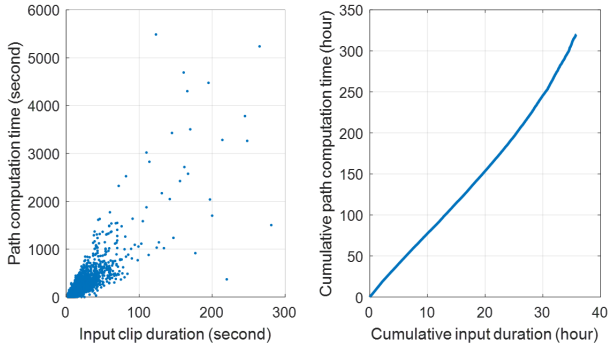
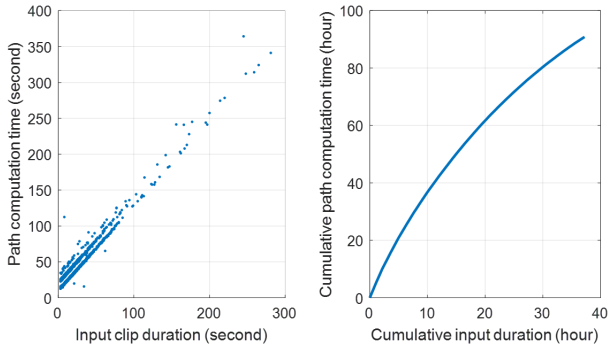


Figure 15: *Representative results of the path computation and 3D reconstruction with Metashape.*



(a) PhotoScan



(b) ORB-SLAM2

Figure 16: *Computation time of automatic annotations by (a) Metashape and (b) ORB-SLAM2. (left): A scattered plot of a tendency between input clip duration and path computation time. (right): The tendency between the cumulative input duration and the cumulative path computation time is approximately linear.*

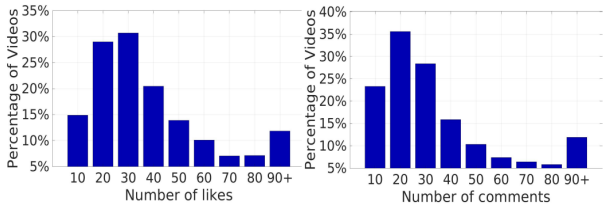


Figure 17: *Distribution of the number of likes (left) and comments (right).*

Table 4: *Evaluation metrics for the classification of time information and scene type for each class and by different approaches.*

Type	Method		Precision	Recall	F-score
Time: day/night/between	SVM	day	0.42	0.28	0.34
		between	0.73	0.57	0.64
		night	0.42	0.89	0.57
	2D CNN	day	**0.75**	0.83	0.79
		between	0.81	**0.73**	**0.77**
		night	0.89	0.89	0.89
	(2+1)D CNN	day	0.72	**0.83**	**0.79**
		between	**0.90**	0.66	0.76
		night	**0.90**	**0.96**	**0.93**
Scene type: urban/rural/mixed	SVM	rural	0.45	0.18	0.26
		mixed	0.43	**0.91**	**0.58**
		urban	0.62	0.13	0.22
	2D CNN	rural	0.64	0.70	0.67
		mixed	0.56	0.50	0.53
		urban	**0.64**	0.65	0.64
	(2+1)D CNN	rural	**0.68**	**0.80**	**0.74**
		mixed	**0.62**	0.44	0.52
		urban	0.63	**0.74**	**0.68**

Table 5: *Evaluation metrics for our classifiers. $*$ denotes that we compute the mean value of the precision, recall, and F-Score for the multi-label classifications.*

Type	Method	Precision	Recall	F-score
Clip category: drone/non-drone	SVM	0.67	0.67	0.68
	2D CNN	**0.90**	0.78	0.84
	(2+1)D CNN	0.86	**0.87**	**0.86**
Logo: yes/no	SVM	0.52	0.55	0.53
	2D CNN	0.73	0.75	0.74
	(2+1)D CNN	**0.79**	**0.87**	**0.83**
Location: indoor/outdoor	SVM	**0.86**	0.64	0.73
	2D CNN	0.79	0.93	0.85
	(2+1)D CNN	0.81	0.93	**0.87**
	Places365-CNN	0.71	**0.97**	0.82
Time*: day/night/between	SVM	0.52	0.58	0.52
	2D CNN	0.82	0.82	0.82
	(2+1)D CNN	**0.84**	**0.82**	**0.83**
Scene type*: urban/rural/mixed	SVM	0.5	0.41	0.35
	2D CNN	0.61	0.62	0.61
	(2+1)D CNN	**0.64**	**0.66**	**0.65**