Learning from PhotoShop Operation Videos: The PSOV Dataset

Jingchun Cheng[2], Han-Kai Hsu[1], Chen Fang[3], Hailin Jin[3], Shengjin Wang[2](✉), and Ming-Hsuan Yang[1]

[1] University of California, Merced, Merced, USA
[2] Tsinghua University, Beijing, China
wgsgj@tsinghua.edu.cn
[3] Adobe Research, San Jose, USA

Abstract. In this paper, we present the PhotoShop Operation Video (PSOV) dataset, a large-scale, densely annotated video database designed for the development of software intelligence. The PSOV dataset consists of 564 densely-annotated videos for Photoshop operations, covering more than 500 commonly used commands in the Photoshop software. Videos in this dataset are obtained from YouTube, manually watched and annotated precisely to seconds by experts. There are more than 74 h of videos with 29,204 labeled commands. To the best of our knowledge, the PSOV dataset is the first large-scale software operation video database with high-resolution frames and dense annotations. We believe that this dataset can help advance the development of intelligent software, and has extensive application aspects. In this paper, we describe the dataset construction procedure, data attributes, proposed tasks and their corresponding evaluation metrics. To demonstrate that the PSOV dataset has sufficient data and labeling for data-driven methods, we develop a deep learning based algorithm for the command classification task. We also carry out experiments and analysis with the proposed method to encourage better understanding and usage of the PSOV dataset.

Keywords: Software intelligence · The PSOV dataset · Photoshop operation video

1 Introduction

Recent years have witnessed rapid development in software intelligence. With the performance leap made by deep learning, there is an explosion of works in automatic human-assisting techniques, e.g. advanced driver assistance system [4,5,22,28], machine translation [2,8], interactive robots [21,26,36], and

J. Cheng and H.-K. Hsu—Equal contribution.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-20870-7_14) contains supplementary material, which is available to authorized users.

virtual player [14,15,30,40,41]. As many state-of-the-art algorithms are data-driven, well-designed datasets [1,11,29] contribute a lot to this prosperity. For example, [9,12,23] boost the development in image classification, [13,34] enable rapid progress in robotic vision; [19,27,33] assist researches in action recognition largely. In spite of the numerous existing datasets, there is still a lack of data for one particular use: computer software intelligence.

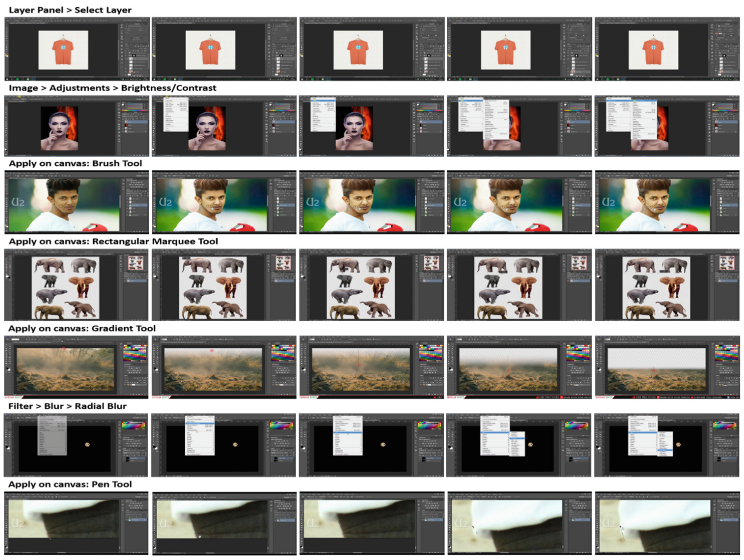

Layer Panel > Select Layer
Image > Adjustments > Brightness/Contrast
Apply on canvas: Brush Tool
Apply on canvas: Rectangular Marquee Tool
Apply on canvas: Gradient Tool
Filter > Blur > Radial Blur
Apply on canvas: Pen Tool

Fig. 1. Example frames of the PSOV dataset. Each row represents a video clip for one specific Photoshop operation.

Computer software plays an important role in everyday life. Due to the never-satiable appetite of computer users, there exist an rapidly-growing number of computer software, varying a lot in function, operation, and etc. Therefore, it is important for software to provide easy access for beginners. A common solution is to put all technical details in a user-guide. However, the long, boring, rich-text user-guide itself causes trouble for starters. We propose that software intelligence could be incorporated here to help solve this problem. Starters of a software can first refer to an intelligent agent, which briefly narrates shared Internet instructional videos, and advises users with instructional videos in correspondence with specific needs. With the help of these highly related, readily comprehensible instruction videos recommended by intelligent software agents, the software can be much easier to understand, operate and spread.

We consider software intelligence as a next research hotspot due to the predictable huge potentials. However, there are few published datasets designed to help algorithms understand software operations. The most closely related dataset, MiniWoB [29], aims to provide simulated environment and data helping software agents to learn interactive tasks on the web. But this dataset only uses synthetic video data, which has small window size (160×210 pixels), simple operations and primitive interfaces. Motivated by this observation, we propose to construct a computer software dataset that can further encourage research and development of intelligent software in real-world situations. For concreteness, we first focus on one widely-used software with hundreds of complex operations, Photoshop. We collect a large number of Photoshop operation videos (mostly instructional videos), annotate them and also propose some tasks for easy entry.

In this paper, we present a large-scale, densely-annotated PhotoShop Operation Video (PSOV) dataset (Fig. 1). The PSOV dataset contains videos and dense command annotations for real-world Photoshop Software. Each annotation includes command name, start and end time accurate to seconds. There are 74 h of videos and 29,204 labeled commands in the dataset. In addition, we define three tasks on the PSOV dataset: command classification, command tube prediction, and command recognition. The details of each task and evaluation metrics are described in Sect. 3. To have a better insight of the proposed dataset, we also construct a 3-D convolutional neural network based algorithm for the command classification task. By experimenting with the proposed network, we validate that the PSOV dataset is capable of supporting deep learning methods, and encourage further understanding of this database. The dataset, task definition, evaluation code as well as annotation tool are available at http://vllab1.ucmerced.edu/~hhsu22/PSOV/.

The main contributions of this work are: (1) a first-of-its-kind, large-scale, real-world PSOV dataset containing dense command annotations; (2) three well-designed tasks with evaluation metrics to help develop software intelligence; (3) a baseline algorithm for better usage and comprehension of the proposed dataset.

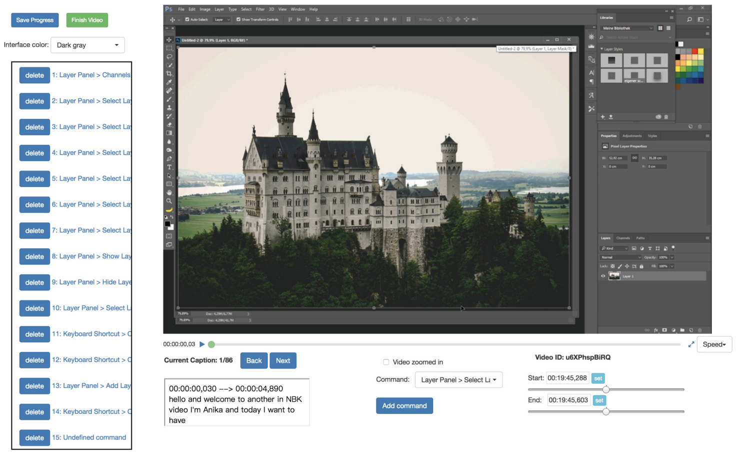
2 Dataset Construction Procedure

Raw videos for Photoshop operations are downloaded from YouTube[1]. The videos are collected using the Youtube Data API[2], which allows users to search for corresponding video information (such as video title, views, likes, and duration) using keywords We use keywords like *Photoshop*, *Photoshop Introduction*, *Photoshop Operation* and *Photoshop Tutorial* to search for potential Photoshop videos. The API does not return all the related videos on Youtube due to some restrictions. In order to look for as many videos related to the given keyword as possible, we set different time windows and make multiple searches for each keyword. We take the union of all the search results and remove duplicate videos programmatically. We also filter these videos with the requirement of a minimum 720p resolution. This procedure results in a collection of 184,626 videos. We observe that videos which are more related to Photoshop operations often have some creator input, i.e. caption data in the video metadata file. To guarantee high quality, we only keep the videos which have caption data, resulting in 3,734 remaining videos. Then, we go through captions of each video and sort out

[1] www.youtube.com.
[2] developers.google.com/youtube/v3/.

more than 2,000 low-quality or non-related ones. Finally, each video is watched and evaluated manually until we reach the final 564 high-quality Photoshop operation videos.

For labeling, we use a crowdsourcing platform due to the huge amount of this work. We annotate each and every command performed in the collected videos with the help of several workers, who has experience in using Photoshop. The workers are hired from Upwork[3], a global freelancing platform which enables remote communication and collaboration. Upwork provides the option to specify skill-level requirements for tasks, allowing us to hire workers with a certain level of Photoshop software knowledge. Other than their Photoshop skills, we also set a rating requirement for workers (each worker has an averaged rating from their previous jobs). Only the top-ranked workers in the platform are invited for our labeling project. Before labeling, we also conduct an interview and a training process to ensure the qualification of the workers.



Interface color: Dark gray

1: Layer Panel > Channels
2: Layer Panel > Select La
3: Layer Panel > Select La
4: Layer Panel > Select La
5: Layer Panel > Select La
6: Layer Panel > Select La
7: Layer Panel > Select La
8: Layer Panel > Show Lay
9: Layer Panel > Hide Laye
10: Layer Panel > Select L
11: Keyboard Shortcut > C
12: Keyboard Shortcut > C
13: Layer Panel > Add Lay
14: Keyboard Shortcut > C
15: Undefined command

00:00:00,03

Current Caption: 1/86

00:00:00,030 --> 00:00:04,890
hello and welcome to another in NBK video I'm Anika and today I want to have

Video zoomed in

Command: Layer Panel > Select La

Video ID: u6XPhspBiRQ

Start: 00:19:45,288

End: 00:19:45,603

Fig. 2. Labeling tool for Photoshop videos. This figure shows an example interface of our online labeling tool, by which workers can easily annotate a Photoshop operation video. Past annotations are listed on the left. See detailed descriptions in Sect. 2.

During the annotation process, we implement an online annotation tool to help facilitate remote working as well as simplify the labeling process (released with the dataset). Figure 2 shows an interface of the annotation tool, a web application based on the Express.js framework[4]. The tool shows a progress bar for each video, enabling workers to easily navigate through a video and precisely locate commands. In the bottom-right corner, there are two additional time bars designed for fine adjustment of the start and end time point, respectively. These progress bars each represents a 20-s interval with the selected time point in

[3] www.upwork.com.
[4] www.expressjs.com.

the center. They contribute a lot to time precision during annotation. Workers can select an approximate time on the full-length progress bar, and make small adjustments here to be accurate to seconds. Users can change the video playback speed (next to the full-length progress bar) in case that the candidate video is fast forwarded by its creator. We also consider other factors like interface color which differs due to software versions or themes, and video zoom-in selection when the software does not occupy the full screen (e.g. the bottom row in Fig. 6).

Before starting the labeling process, each worker is assigned an account name for user identification. Workers need to log in to their own account to start labeling. Videos are assigned randomly to workers one at a time, with the text input from video creator as a reference during labeling (see the text box in the bottom-left of Fig. 2). Workers are allowed to add, delete, and insert command labels. The labeling process also requires the user to select an interface color of the Software. In the labeling process, we set these two pre-defined colors for workers to choose from (dark gray and light gray). The most important and time-consuming part is for workers is to label the start and finish time, as well as the specific operation name of each command. Along with that, workers are required to judge whether the entire software interface is within the screen during each command, because videos may be post-edited by creators to zoom into a specific region in some cases, and video frames vary a lot after zooming in (see bottom row in Fig. 6 for an example). After labeling the candidate video, workers can either click *Finish Video* to receive the next assignment and upload the current one onto the server, or click *Save Progress* and return to where they left off afterwards. Finally, the labeled commands are double checked by ourselves to ensure the correctness.

We show the pipeline of our dataset construction procedure in Fig. 3, and describe details of the PSOV dataset in Sect. 3.

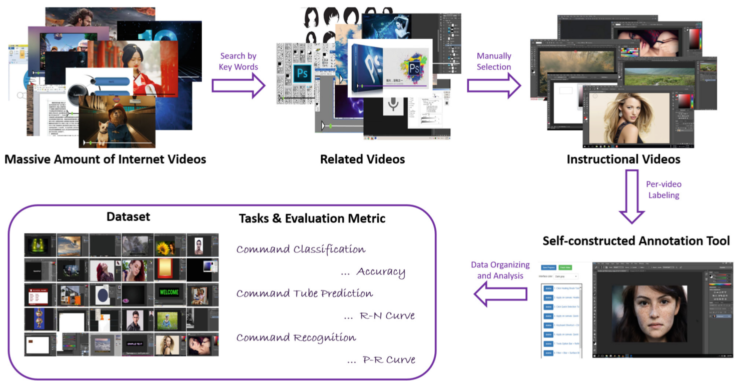


Fig. 3. Pipeline of dataset construction procedure.

3 Dataset Description

In this Section, we introduce the PSOV dataset, a large-scale, densely-annotated video dataset, specially designed for development of software intelligence. Example frames of some typical Photoshop commands are shown in Fig. 1; and the dataset structure is illustrated in Fig. 5.

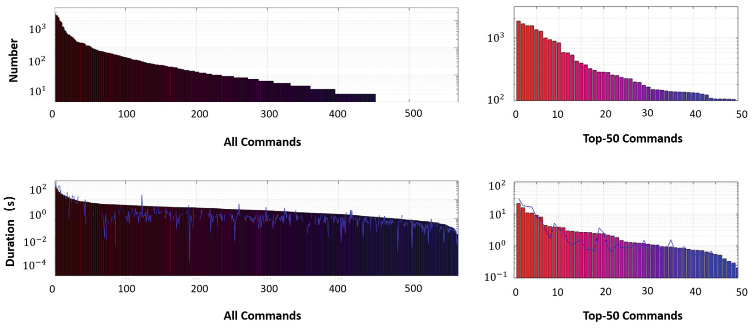


Fig. 4. Data distribution in the PSOV dataset. From left to right, we show data distribution of all labeled commands and the top-50. The first row shows the number of each command, and the second row shows the average duration (finish time minus start time) of the commands, all sorted by value. The blue lines in duration figures denote duration variance values of each bar. The top-left figure shows that about 150 commands only have 1 labeled sequence, and that the top-50 commands have the number of sequences more than 100. We provide labeling data for all command sequences in the dataset, but only evaluate tasks on the top-50 to guarantee enough training data. More details will be presented in the supplementary materials. (Color figure online)

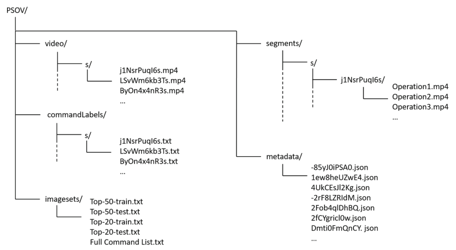


Fig. 5. Dataset structure. This figure illustrates the data storage path in the dataset, where folder *video*, *commandLabels*, *segments*, *metadata* contains whole videos, per-video annotation file, operation frames, and video caption information respectively.

Data Amount and Quality. The PSOV dataset consists of 564 densely-annotated Photoshop operation videos. There are 74 h of video with 29,204 labeled commands. Each video has the minimum resolution of 720p. Labels of command operations in Photoshop are predefined by ourselves by exploiting user guides, technique books, etc. The command definition is in a concise and effective manner, for instance, *Layer Panel > Select Layer*, *Image > Adjustments > Brightness/Contrast*, and *Apply on canvas: Brush Tool*. All 29,204 commands are labeled by the workers hired from Upwork (see details in Sect. 2). Note that besides a large portion of usual mouse click interactions, the labels also include keyboard short-cuts (e.g. Control-N, Control-C, Control-P) which are often used in Photoshop software. These keyboard short-cuts make the dataset more challenging and more realistic, since they are hard to recognize for beginners. The number of samples for each command is shown in Fig. 4. We select the top-50 commands (those with larger amount) for tasks on this dataset to ensure a sufficient volume of data for data-driven techniques like CNN-based video processing algorithms [18,19,24,31,42]. In Sect. 6, we demonstrate that the PSOV dataset has enough data quantity and diversity for training deep-learning based algorithms.

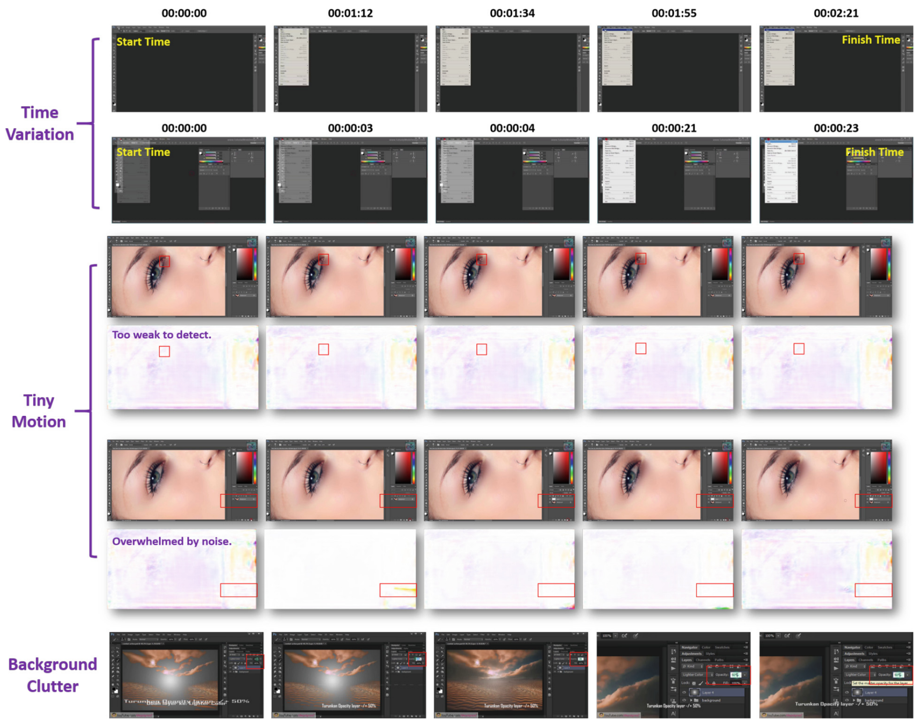


Fig. 6. Key difficulties in the PSOV dataset. This figure shows examples of three key difficulties in the PSOV dataset, i.e. duration variance, tiny motion and background clutter. (1) For time variance, each row shows a sequence of class *File > New*. This same operation has a two-second difference in the two sequences. (2) For tiny movement, optical flow [10] is used to illustrate what can be seen in motion space: the brush moving around lower eyelid in the first sequence is too weak to be detected; and the new layer appeared in the right panel has no stronger response than background noises. (3) For background clutter, we show an extreme example where the center region is zoomed in by the creator, largely changing the background of the operation. (Color figure online)

Challenges. The PSOV dataset is a challenging dataset, as the video data collected have fast but minor motion, with large variance in duration and background (see Fig. 6 for an example). In addition, data samples of different commands are imbalanced. Figure 4 presents an example of the duration difference between two command sequences with the same label *File* > *New*, showing a key difficulty of time variance in this dataset. The figure also gives examples of two other challenges: minor motion and background clutter. The PSOV dataset holds many sequences where motion happens in a tiny local area that can not be distinguished by current optical flow methods [10,17,25,35]. Furthermore, some operation sequence may contain severe background clutter (zooming-in, panel change, etc.), causing confusion for recognition. We analyze the influence of some challenging factors using the proposed method in Sect. 6, providing a better insight and understanding of the PSOV dataset.

4 Tasks and Evaluation

In this section, we describe three tasks as well as the corresponding evaluation metrics on the PSOV dataset. Tasks share common training and testing set, which contains 433 and 131 videos, respectively (training and testing sets are split in a manner that the command distributions are similar). To ensure a sufficient amount of labeling data for deep learning methods, we only conduct these tasks on the top-50 and top-20 popular commands. We also provide the evaluation functions in the development kit to release together with the dataset (see supplementary for details).

Command Classification. The command classification task aims to recognize the operation performed in a given video tube[5]. In this task, the start and finish time of commands in both training and testing sets are given. Algorithms need to learn a classifier from the 433 training set videos and predict the command label for the operation tubes in the test set. We use the simple and intuitive classification accuracy to evaluate the performance of different methods on this task. Using our development kit, the classification accuracy and per-class precision will be given once obtaining a 50-dimension probability vector from the algorithm.

Command Tube Prediction. This task aims to predict the begin and finish time of each command in Photoshop videos. With the available training set videos and corresponding command labels, methods need to predict the two time points (start time t_{start} and finish time t_{end}) for each operation in the test set videos. We propose to use a R&N Curve[6] as the evaluation metric, where a tube is considered 'hit' when a proposal has the IoU (intersection over union between ground truth time interval and the proposal) greater than 0.5. Note that commands not in the top-50 are not calculated in this task. The methods on this

[5] Video tube denotes a sequence of video frames which contains one specific command.

[6] R denotes recall, and N denotes the number of proposals averaged over the number of ground truth commands.

task predict proposals for command tubes ($[t_{start}, t_{end}]$); and are evaluated by R&N Curve where higher curves denote better performance.

Command Recognition. Command recognition is a comprehensive and the most complicated one among three tasks. This task is a further step from classification and tube prediction, it aims to recognize commands (predict start time, end time, command name of operations) from a raw video. Given a test video, the algorithm needs to decide which time period exists an operation and exactly which one it is. It is closest to reality, as the method understands when and what commands are performed in videos with no manually provided information. Algorithms for this task can be directly applied to Photoshop operation videos outside of the PSOV dataset. They can sketch instructional videos with step by step operation list, relieving users from browsing over tens of thousands of video searching results. They can also dig useful data from massive amount of videos uploaded to the Internet every day, and provide assistance to researchers and software developers. Furthermore, their output operations can also be transformed back into computer commands, so that the computer can reproduce automatically in real-world software. The command recognition task is evaluated by AUC (the Area Under precision-recall Curve). Note that the correct prediction here has an IoU with the ground truth over a certain threshold and a correct command label prediction. Both precision-recall curve and AUC value are provided by development kit in evaluation.

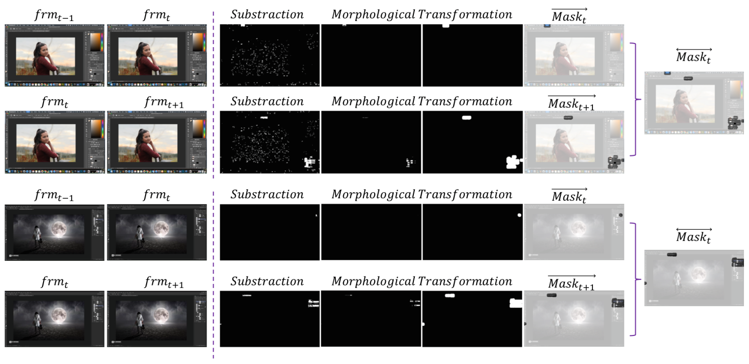
5 Methodology

We develop a command classification algorithm on the PSOV dataset to: (1) show an example usage of the proposed database; (2) validate that the PSOV dataset has sufficient data volume for developing data-driven algorithms; and (3) provide a baseline comparison for the command classification task. This section describes the details of our method construction.

Convolutional neural network (CNN) plays an important role in computer vision these years for the effectiveness and robustness of CNN features and classifiers. Many algorithms [3,6,16,42] use CNN for video recognition, which usually process each frame independently and use feature fusion to obtain video descriptions. However, such methods make little use of the motion information in time dimension since the feature of each frame is extracted separately. In this paper, we propose to use a 3-D CNN [18,37] for the challenging PSOV dataset. First, we design an attention-aware preprocessing method to draw attention to operation-critical regions. Then we regularize each video to a fixed length with reference to the attention information. Finally, a 3-D CNN structure is trained for the command classification task.

5.1 Attention-Aware Filtering

In video-related tasks, it is common to leverage temporal features [7,32]. Optical flow is one of the most commonly used descriptors for such information. However,



frm_{t-1}
frm_t
Substraction
Morphological Transformation
$\overrightarrow{Mask_t}$
frm_{t+1}
$\overrightarrow{Mask_{t+1}}$
$\overleftrightarrow{Mask_t}$

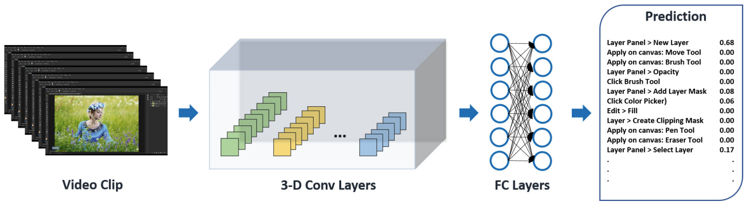
Fig. 7. Attention-aware filtering. We show the attention filtering results on example frames from two commands: *Layer Panel > Select Layer* and *Layer Panel > Duplicate Layer*. $\overrightarrow{Mask}$ denotes the attention computing in single direction ($frm_{t-1} \Rightarrow frm_t$, $frm_t \Rightarrow frm_{t+1}$); and $\overleftrightarrow{Mask}$ denotes the result considering bi-directional context ($frm_{t-1} \Rightarrow frm_t \Leftarrow frm_{t+1}$).

this traditional feature does not take effect in the PSOV dataset, for that key motions are usually weak or located in a tiny region in this dataset (Fig. 6). If we use the whole image frame as network input, information from key regions can easily be overwhelmed by surrounding background noises. Thus, we propose an Attention-aware Filtering algorithm that directly extracts features from the strongest motion part by filtering out useless area. Figure 7 shows an example of the process for our filtering method, which helps the network to focus more on the informative and effective region, boosting its recognition ability (see Sect. 6.1).

Difference Filtering. The purpose of our Attention-aware Filtering method is to focus on the informative motion region. As shown in Fig. 6, the motion that determines specific operation often takes place in a small fraction of area. An intuitive way to find this area is to use the difference map between two adjacent frames ($frm_t - frm_{t-1}$). However, due to video compression artifacts, the direct subtraction has a noisy result (column three in Fig. 6), making it difficult to locate true movement. To deal with this phenomenon, we propose to use morphological image processing methods: erosion and dilation. First, we apply erosion with a disk-shaped kernel (radius 1 pixel) on the subtraction result, removing noisy points here and there. As the erosion procedure comes with region shrinking, we then apply a dilation kernel (this time by a disk-shaped kernel with a radius of 20 pixels) to ensure that most information remains in the outcoming mask ($\overrightarrow{Mask_t}$). Figure 7, shows that Difference Filtering can effectively locate main movement region, relieving the difficulty caused by minor motion in the PSOV dataset.

Bi-direction Context. As movement happens between two frames, context information is needed for both before and after the action ($frm_{t-1} - frm_{t+1}$). While $\overrightarrow{Mask_t}$ is calculated between two frames, it only knows what happened before the action but has no idea about the temporal context afterward. This can cause serious information loss. For example, command *Layer Panel > Select Layer* and *Layer Panel > Duplicate Layer* share similar actions in the former part of the operation in Fig. 7. Simply using Difference Filtering leads to a confusion on $\overrightarrow{Mask}$ in the first and third row. Based on this observation, we propose to compute bi-directional context that preserves temporal context information both before and after the current frame. As shown in Fig. 7, $\overrightarrow{Mask_t}$ and $\overrightarrow{Mask_{t+1}}$ are obtained using Difference Filtering introduced above. These two masks are combined together to obtain a $\overleftrightarrow{Mask}$ which preserves bi-direction temporal information. We show the classification difference with and without bi-direction context in Fig. 10, where class 0 and 12 denotes class *Layer Panel > Select Layer* and *Layer Panel > Duplicate Layer* respectively. The left side figure shows that a large proportion of class 12 video clips are miss-categorized into class 0 due to information loss caused by Difference Filtering. The right-side figure demonstrates that adding bi-direction context can effectively relieve this problem.

As described above, our Attention-aware Filtering uses Difference Filtering with Bi-direction Context. It can find the main active region as well as examine the temporal context in both forward and backward direction. We demonstrate in Sect. 6.1 that this process is a significant step in the proposed algorithm.



Layer Panel > New Layer	0.68
Apply on canvas: Move Tool	0.00
Apply on canvas: Brush Tool	0.00
Layer Panel > Opacity	0.00
Click Brush Tool	0.00
Layer Panel > Add Layer Mask	0.08
Click Color Picker)	0.06
Edit > Fill	0.00
Layer > Create Clipping Mask	0.00
Apply on canvas: Pen Tool	0.00
Apply on canvas: Eraser Tool	0.00
Layer Panel > Select Layer	0.17
.	.
.	.
.	.

Fig. 8. Network structure.

5.2 Video Regularization

All the video sequences are processed to the same size (in both spatial and temporal dimension) for convenience during training and testing in the proposed framework. Unlike images which can easily be resized to a fixed size, videos vary a lot in temporal extent (the first row in Fig. 6), especially for the proposed PSOV dataset. To regularize the Photoshop command video clips, one significant point is to select frames which keep the most important information. We note that simple uniform sampling can miss such important information severely in the PSOV dataset, for that the key moments (frames that determines the command like clicking a button) distribute randomly in each video. Therefore, we take advantage of the attention area in Sect. 5.1. Redundant frames with no information left after attention filtering are removed from the video. During down-sampling, we start by taking away frames with less information; when it comes to up-sampling, we simply pad the video via random duplicating.

5.3 3-D CNN

Our network is modified from [38], with 5 convolution layers, two fully connected layers, and one classification output layer. Figure 8 shows the structure of our proposed network. Different from 2-D CNNs, 3-D network does convolution and pooling in 2-D surface and an additional time dimension, so sizes and strides in this network have three parameters: width, height and time. Our detailed parameter settings are as follow: (1) convolution layers all have $3 \times 3 \times 3$ kernels with $2 \times 2 \times 2$ strides; (2) pooling layers have $2 \times 2 \times 2$ kernels with strides of $2 \times 2 \times 2$ (except the first pooling layer which has a $2 \times 2 \times 1$ stride, with no operations in temporal channel); (3) the two fully connected layers both have 2,048 output channels and are followed by drop-out layers.

Table 1. Ablation study. This table shows the performance for different combinations of components in the proposed framework, where 3D-Conv, Diff-Filter, Bi-Context, DataAug denote 3-D convolution, Difference Filtering, Bi-direction context, data augmentation respectively (details in Sect. 6.1).

	3-D CNN	Diff-Filter	Bi-Context	DataAug	acc-50	acc-20
2-D CNN		✓			17.02	23.18
2-D CNN*		✓	✓		19.15	24.70
RGB input	✓				51.17	54.66
3-D CNN	✓	✓			63.15	69.14
3-D CNN*	✓	✓	✓		63.76	69.73
Ours	✓	✓	✓	✓	66.37	74.97

The training process is done with Pytorch on a 12G TitanX GPU. The proposed network is trained from scratch, using SGD optimizer with learning rate of 1e−5, and momentum of 0.9. Videos are regularized to 100×100 pixels in spatial domain and 50 frames in the temporal domain. It takes about 250 epochs to reach convergence with batch size of 10. Evaluation of the proposed network can be found in Sect. 6.

5.4 Data Augmentation

Data augmentation is widely used in various computer vision fields [10,20,39]. It can help introduce more diversity and make up for the data imbalance among different classes (Fig. 4). Although the PSOV is a large dataset, data augmentation is still helpful for network training. We use the following sets of data augmentation methods to augment training data: (1) image enhancement, where we adjust the brightness, saturation, contrast, and sharpness of each video frame to augment training data, (frames within one command period have the same augmentation setting); (2) noise, where we randomly add two kinds of noises to each frame during training: the Gaussian white noise and the salt and pepper noise; (3) translation, where we add a bit of movement to frames and use neighboring pixels to compensate for the corresponding blank area.

6 Experiments

We carry out extensive experiments and analysis on the PSOV dataset. First, we validate the effectiveness of each method component. Then we test the influence of command duration variance, filtered area, and analyze the confusion matrix. Through these experiments, we demonstrate that the PSOV dataset is sufficient to support deep learning, and hope to encourage better understanding and usage of this dataset.

6.1 Ablation Study

First, we validate the necessity and effectiveness of each component in the proposed algorithm via ablation study on the PSOV dataset. The proposed framework mainly have the following components:

- **3D-Conv**, 3-D Convolution, without which network does calculations in space domain only without temporal dimension;
- **Diff-Filter**, Difference Filtering, primary step in Attention-aware Filtering, without which network takes in original RGB images;
- **Bi-Context**, Bi-direction Context, without which network only uses Difference Filtering (in Sect. 5);
- **DataAug**, Data Augmentation, without which network does not use data augmentation during training.

We implement methods with different sets of components, and compare their performance for classifying top-20 and top-50 commands respectively. Note for top-20 command classification, networks are trained with top-20 classes on the training set; while for top-50 classification, networks are trained with top-50 classes.

Table 1 shows the statistical results, where components used in each method are denoted by check-marks. We observe that 3-D Convolution contributes largely (3-D CNN vs 2-D CNN), improving the accuracy by more than 40%. It demonstrates that temporal information is essential in recognizing Photoshop operations. The Difference Filtering (Diff-Filter) and Bi-direction Context (Bi-Context) also consistently improve performance by about 10% and 1%, illustrating the effectiveness of our Attention-aware Filtering step (Diff-Filter+Bi-Context). We also evaluate the data augmentation (DataAug) step, and find a 2%–5% improvement (3D CNN* vs Ours) in top-50 and top-20 respectively, proving that data augmentation helps on the PSOV dataset.

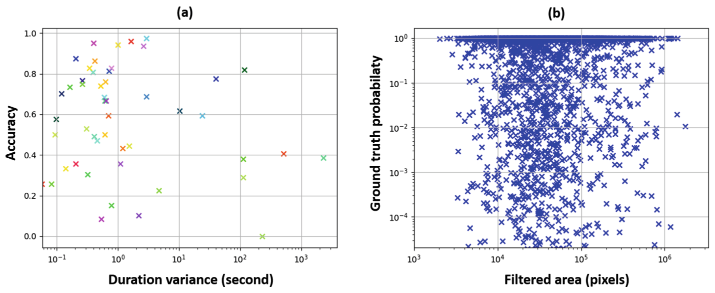


Fig. 9. Accuracy distribution. Image (a) shows the per-class accuracy distribution on training set class duration variance; image (b) shows the per-sequence prediction versus averaged attention filter area curve. See details in Sect. 6.2.

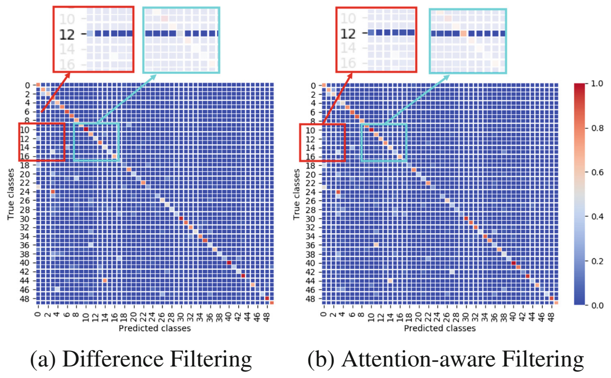


Fig. 10. Confusion matrix for Difference Filtering and Attention-aware Filtering. Index of each row and column denotes a top-50 class; color blue to red in each pixel indicates the proportion of its row-class classified into column-class; class 0 and 12 are *Layer Panel > Select Layer* and *Layer Panel > Duplicate Layer* respectively. (Color figure online)

6.2 Analysis on the Command Classification Task

We draw the figure of per-class accuracy versus class duration variance in training set (image (a) in Fig. 9). This figure shows that command sequences with extremely large duration variance (right side of the image) tend to be hard to classify (the five points in the bottom-right corner; while other videos with smaller duration variance (<1 s) do not have clear correspondence between accuracy and variance. It demonstrates that the proposed network can handle time variance in the PSOV dataset to a large extent.

We also draw the true class prediction probability of each sequence with the averaged pixel number of our Attention-aware Filtering to see whether the filtered motion area is related to classification difficulty. Image (b) in Fig. 9 shows the results of this distribution, illustrating that the proposed algorithm is robust to motion area.

Figure 10 shows two confusion matrix for with and without Bi-direction Context, respectively. Index of each row or column represents a top-50 class. Color blue to red in each pixel indicates the proportion of its row-class classified into column-class. Figure 10 shows an intuitive inter-class similarity (similar pairs like class 0 and 12, class 14 and 44), indicating that bi-direction context helps in correcting wrong predictions. For example, class 0 (*Layer Panel* > *Select Layer*) and class 12 (*Layer Panel* > *Duplicate Layer*) are largely misclassified with using only Difference Filtering, but the miss-classifications are corrected via adding bi-direction context (the Attention-aware Filtering). Detailed explanation of how this happens is in Sect. 5.1 and Fig. 7.

7 Conclusion

In this paper, we present the PSOV dataset, a novel, large-scale, densely-annotated, Photoshop Operation Video dataset. The PSOV dataset consists of 564 videos with 29,204 dense annotations. To the best of our knowledge, it is the first real-world software operation dataset with large amount of videos and detailed labeling. We believe that this database can fuel researches in software intelligence, e.g. instruction video mining, autonomous software component, etc. To have a better insight into the PSOV dataset, we also propose a baseline algorithm for the command classification task. By experimenting with the proposed framework, we (1) validate that the PSOV dataset has sufficient data quantity for deep learning, (2) evaluate the effectiveness of each algorithm component, and (3) encourage better understanding and usage of the database. In the future, we plan on extending our dataset to more popular software, and provide online challenges.

Acknowledgement. This work is supported in part by the NSF CAREER Grant #1149783, and gifts from Adobe.

References

1. Abbeel, P., Coates, A., Ng, A.Y.: Autonomous helicopter aerobatics through apprenticeship learning. IJRR **29**(13), 1608–1639 (2010)
2. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
3. Caelles, S., Maninis, K.K., Pont-Tuset, J., Leal-Taixé, L., Cremers, D., Van Gool, L.: One-shot video object segmentation. In: CVPR (2017)
4. Chen, C., Seff, A., Kornhauser, A., Xiao, J.: DeepDriving: learning affordance for direct perception in autonomous driving. In: ICCV (2015)
5. Chen, X., Ma, H., Wan, J., Li, B., Xia, T.: Multi-view 3D object detection network for autonomous driving. In: CVPR (2017)
6. Cheng, J., et al.: Learning to segment instances in videos with spatial propagation network. arXiv preprint arXiv:1709.04609 (2017)
7. Cheng, J., Tsai, Y.H., Wang, S., Yang, M.H.: SegFlow: joint learning for video object segmentation and optical flow. In: ICCV (2017)

8. Cho, K., et al.: Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078 (2014)
9. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: a large-scale hierarchical image database. In: CVPR, pp. 248–255 (2009)
10. Dosovitskiy, A., et al.: FlowNet: learning optical flow with convolutional networks. In: ICCV (2015)
11. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: an open urban driving simulator. In: CoRL (2017)
12. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The Pascal visual object classes (VOC) challenge. IJCV **88**(2), 303–338 (2010)
13. Geiger, A., Lenz, P., Stiller, C., Urtasun, R.: Vision meets robotics: the KITTI dataset. IJRR **32**(11), 1231–1237 (2013)
14. Gelly, S., Silver, D.: Achieving master level play in 9 × 9 computer go. In: AAAI (2008)
15. Gelly, S., Silver, D.: Monte-Carlo tree search and rapid action value estimation in computer go. Artif. Intill. **175**(11), 1856–1875 (2011)
16. Girdhar, R., Ramanan, D., Gupta, A., Sivic, J., Russell, B.: ActionVLAD: learning spatio-temporal aggregation for action classification. In: CVPR (2017)
17. Ilg, E., Mayer, N., Saikia, T., Keuper, M., Dosovitskiy, A., Brox, T.: FlowNet 2.0: evolution of optical flow estimation with deep networks. In: CVPR (2017)
18. Ji, S., Xu, W., Yang, M., Yu, K.: 3D convolutional neural networks for human action recognition. PAMI **35**(1), 495–502 (2013)
19. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: CVPR (2014)
20. Khoreva, A., Benenson, R., Ilg, E., Brox, T., Schiele, B.: Lucid data dreaming for multiple object tracking. arXiv preprint arXiv:1703.09554 (2017)
21. Kim, M., Kim, S., Park, S., Choi, M.T., Kim, M., Gomaa, H.: Service robot for the elderly. RAM **16**(1), 34–45 (2009)
22. Lefèvre, S., Carvalho, A., Gao, Y., Tseng, H.E., Borrelli, F.: Driver models for personalised driving assistance. VSD **53**(12), 1705–1720 (2015)
23. Lin, T.-Y., et al.: Microsoft COCO: common objects in context. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8693, pp. 740–755. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10602-1_48
24. Oh, J., Guo, X., Lee, H., Lewis, R.L., Singh, S.: Action-conditional video prediction using deep networks in atari games. In: NIPS (2015)
25. Revaud, J., Weinzaepfel, P., Harchaoui, Z., Schmid, C.: EpicFlow: edge-preserving interpolation of correspondences for optical flow. In: CVPR (2015)
26. Rhee, C., Chung, W., Kim, M., Shim, Y., Lee, H.: Door opening control using the multi-fingered robotic hand for the indoor service robot. In: ICRA (2004)
27. Rodriguez, M.D., Ahmed, J., Shah, M.: Action mach a spatio-temporal maximum average correlation height filter for action recognition. In: CVPR (2008)
28. Shashua, A., Gdalyahu, Y., Hayun, G.: Pedestrian detection for driving assistance systems: single-frame classification and system level performance. In: IEEE Intelligent Vehicles Symposium, 2004, pp. 1–6. IEEE, June 2004
29. Shi, T., Karpathy, A., Fan, L., Hernandez, J., Liang, P.: World of bits: an open-domain platform for web-based agents. In: ICML (2017)
30. Silver, D., et al.: Mastering the game of go without human knowledge. Nature **550**, 354–359 (2017)
31. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: NIPS (2014)

32. Song, S., Lan, C., Xing, J., Zeng, W., Liu, J.: An end-to-end spatio-temporal attention model for human action recognition from skeleton data. In: AAAI (2017)
33. Soomro, K., Zamir, A.R., Shah, M.: UCF101: a dataset of 101 human actions classes from videos in the wild. CoRR (2012)
34. Sturm, J., Engelhard, N., Endres, F., Burgard, W., Cremers, D.: A benchmark for the evaluation of RGB-D SLAM systems. In: IROS (2012)
35. Sun, D., Yang, X., Liu, M.Y., Kautz, J.: PWC-Net: CNNs for optical flow using pyramid, warping, and cost volume. In: CVPR (2018)
36. Taggart, W., Turkle, S., Kidd, C.D.: An interactive robot in a nursing home: preliminary remarks. In: COGSCI Workshop (2005)
37. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3D convolutional networks. In: ICCV (2015)
38. Varol, G., Laptev, I., Schmid, C.: Long-term temporal convolutions for action recognition. PAMI **40**(6), 1510–1517 (2017)
39. Voigtlaender, P., Leibe, B.: Online adaptation of convolutional neural networks for video object segmentation. arXiv preprint arXiv:1706.09364 (2017)
40. Wang, J., Xiao, C., Zhu, T., Hsueh, C.H., Tseng, W.J., Wu, I.C.: Only-one-victor pattern learning in computer go. IEEE Trans. Comput. Intell. AI Games **9**(1), 88–102 (2017)
41. Yannakakis, G.N.: Game AI revisited. In: CF (2012)
42. Yue-Hei Ng, J., Hausknecht, M., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: deep networks for video classification. In: CVPR (2015)