Mouse Cursor Detection And Tracking In Instructional Videos

Anonymous WACV submission

Paper ID 834

Abstract

Many expert users of particular software, e.g. Adobe Photoshop regularly post instructional videos online, imparting knowledge on how to perform certain tasks. An important feature that can be used to understand the instructor's actions in an instructional video, is to detect and track the mouse cursor throughout the entirety of each video. Despite recent progress of object detection and tracking, identifying the mouse cursor in such videos is a unique and difficult problem since the mouse cursor typically occupies a very small percentage of the entire frame (0.05-1%), exhibits fast movement, and is prone to instant appearance changes and background clutter. In this paper, we propose a novel three-step tracking-by-detection approach for mouse cursor detection and tracking - unsupervised cursor discovery, multi-scale template matching, and optimal spatiotemporal path search. Our approach is completely unsupervised and is able to handle instant appearance changes and fast movements of the mouse cursor. We present evaluations on a dataset of annotated Adobe Photoshop instructional videos, and show that our method beats conventional online tracking methods such as TLD, MIL and CSRT trackers by a large margin. For a more fair comparison, we also compare our results with Faster-RCNN, a deep learning based object detector, and show comparable success rates.

1. Introduction

Object detection and tracking has been an active area of research in computer vision for several decades. However, most object tracking methods are developed with real-world scenarios in mind and track objects such as people, cars and faces. These objects can be viewed from different camera angles and viewpoints, and exhibit gradual pose change. We choose to address the problem of detecting and tracking mouse cursors in instructional videos, a unique problem that differs in several ways from conventional detection and tracking scenarios. This problem is important because it is one of the first few steps in understanding the instructor's actions, which can pave the way for higher level tasks such as video indexing, video recommendation systems and user behavior modeling. Other tasks that can benefit from mouse cursor detection and tracking include command classification, command tube prediction and command recognition [7].

Conventional tracking methods are generally developed to track objects in real-world environments, and suffer from problems such as drastic changes in illumination, occlusions, and incremental pose variations. However, the mouse cursor detection and tracking problem is a small-object tracking problem that suffers from its own set of issues, such as fast motion, background clutter, scale changes, instant appearance changes and effects resulting from window zooms and pans. Some of these challenges are highlighted in Figure 1.

Also, in our problem, the location of a cursor to be tracked should be initialized in the first frame of a video. Object detection is commonly used for this purpose. Deep learning based approaches [29, 14, 28] have achieved significant performance improvement for general object detection. However, they still have difficulty in detecting small objects due to the lack of visual details [6, 10, 12, 20, 22]. The situation could worsen when they are applied to detect mouse cursors in instructional videos, as the mouse cursors usually appear to be very tiny compared to the high resolutions of these videos. Furthermore, it requires a large amount of training data to train a deep convolutional network. It is difficult to collect a set of videos covering all kinds of mouse cursors as they could have large appearance variations.

To address these issue, we propose a method that can discover mouse cursors in an unsupervised manner and robustly track mouse cursors which could exhibit large appearance change and fast movement in instructional videos. Our method consists of three stages, unsupervised cursor discovery, multi-scale template matching and optimal path search. We utilize simple background subtraction and blob detection to obtain potential mouse cursor templates; perform template matching to obtain possible cursor locations and then use a variant of the MaxPath [32] algorithm to obtain the optimal spatiotemporal path of cursor detections

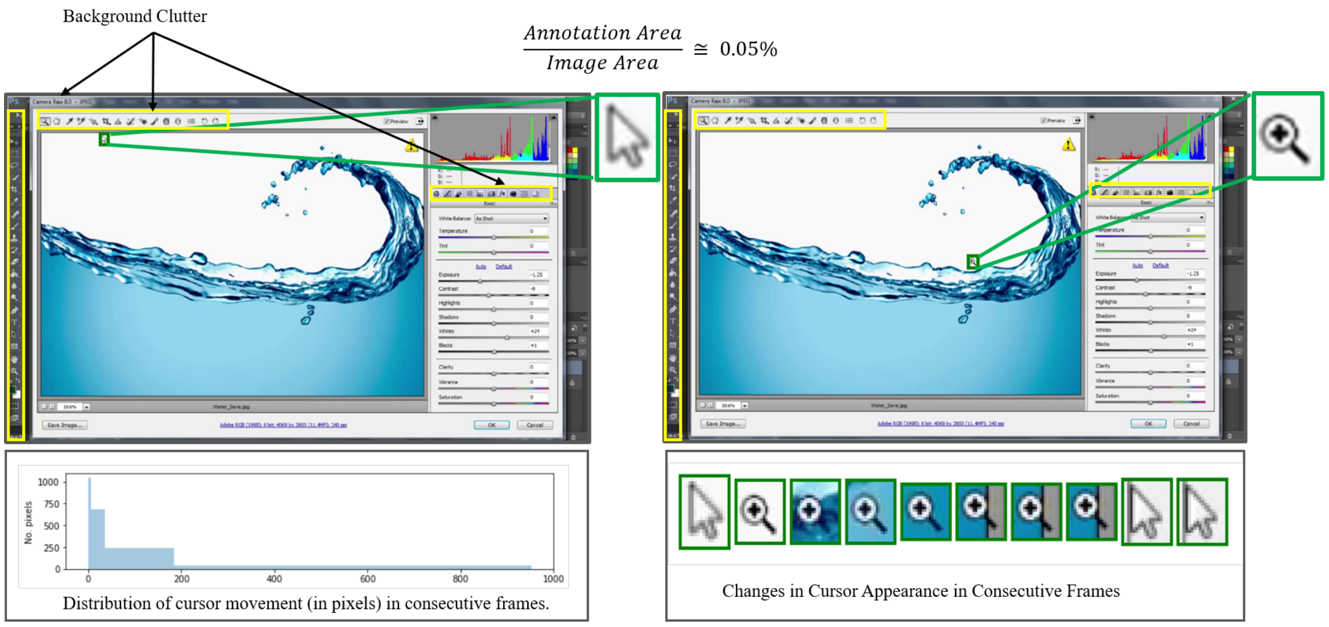
Background Clutter
$\frac{Annotation\ Area}{Image\ Area} \cong 0.05\%$
No. pixels
Distribution of cursor movement (in pixels) in consecutive frames.
Changes in Cursor Appearance in Consecutive Frames

Figure 1. Challenges observed in the detection and tracking of mouse cursors: background clutter in the form other cursor-like objects; small cursors with an average area ratio of 0.05%; fast cursor movements of over 200 pixels observed between consecutive frames in some videos (bottom left); and instant cursor appearance changes between frames (bottom right).

for the entire video. For the purposes of evaluation, we collect a small dataset for this problem consisting of 8 labeled video segments, and compare our approach with online tracking methods (TLD [18], MIL [1], CSRT [24]) and Faster-RCNN, a deep-learning based detection method and summarize the results.

2. Related Work

Template-based tracking is a well-researched problem that dates back to the Lucas-Kanade algorithm [23]. In this type of tracking, a sample image of the target object (a template) is extracted in the first frame, and the image patch closest in similarity with the template is selected in all consecutive frames. Other extensions to this approach have been proposed. [2] deals with parametric transformations of the template and [5], [13] allow for linear appearance changes. Template tracking relies on the assumption that the appearance of the object does not change during the entirety of the video. This assumption fails eventually as the appearance model of the template is no longer similar to the template extracted in the first frame. [25] proposes an update algorithm that does not suffer from template drift, when new templates are extracted at every n^{th} frame. Although our approach is based on template matching, it is quite different from other template-based approaches, because we track a variety of rigid templates after automatically discovering cursors present in a video. Other online tracking approaches, OLB [4], MIL [1] aim to solve the problem by learning to differentiate between the target and background, while trying to avoid the tracker drifting problem; whereas the TLD tracker separates the problem into tracking, learning and detection. Our approach is an offline algorithm and is different from those mentioned above as it draws connections between detections in consecutive frames to determine the path with the highest cumulative template matching score. It is loosely related to the TLD tracker which also uses a detector to correct the tracker when necessary.

Owing to the success of deep convolutional networks (CNNs, [19]) for computer vision tasks, there has been considerable improvement in object detection performance. Despite its challenges, small object detection has seen advances in the past few years. The authors in [22, 6] aim to solve multi-scale object detection by constructing feature pyramids, [20, 12] improve performance further by building top-down architectures with lateral connections, and [10] compensates for the weak semantic information contained in low-level features by fusing high-level features with low-level features via a deconvolution fusion block.

More recently, tracking algorithms that rely on CNNs, MDNet [26], GOTURN [16], SiamFC [3] have become popular. Both GOTURN and SiamFC are Siamese network-based trackers which use a similarity function learned offline, whereas MDNet performs online fine-tuning. As discussed previously, deep learning methods require large amounts of training data, and may have difficulty detecting mouse cursors due to the vast differences between general, real-world objects and mouse cursors. They are also restricted by a predefined set of object labels and cannot discover previously unseen cursors during test time.

Our work is also closely related to unsupervised object discovery in videos. Previous work has tried to address the discovery and localization of objects from weakly-annotated or unlabeled datasets [9, 8, 11, 30, 31], but the task is difficult and most approaches perform worse than strongly supervised methods. Other approaches [17, 27, 33] address similar problems in videos and show that exploiting motion information (absent in static images), may be crucial for weakly supervised object discovery. Whereas previous work deals with general objects that occupy sufficient area in the scene, we are interested in discovering and locating mouse cursors that occupy only 0.05-0.1% of the frame area. For these reasons, we adopt a simple method based on background subtraction and blob detection for unsupervised cursor discovery.

3. Algorithm

Our algorithm takes as input a video and outputs a sequence of mouse cursor locations. It is assumed that in each frame there is at least one mouse cursor. The approach can be divided into three phases - unsupervised cursor discovery, template matching, and optimal path search.

3.1. Unsupervised Cursor Discovery

Successful template matching relies on a template that has a similar appearance and size to the object being matched. Manual collection and annotation of generic cursor templates is an inadequate solution, as cursors exist in a plethora of types and variations, as shown in Figure 4. Moreover, content creators often use custom cursors that are specific to the software they use. To ensure that our technique generalizes well, we need a method to discover cursor templates appearing in a particular video. We achieve this by exploiting unique motion characteristics exhibited by mouse cursors. Instructional videos are prone to background changes from events such as pop-up windows, drop-down menus and other events on the screen. However, they also contain temporal sections in which the background remains unchanged except for cursor motion. Potential templates are obtained by observing such temporal sections.

We denote a frame sequence by $S = \{F_1, F_2, ..., F_N\}$, where F_i is the i^{th} frame in the sequence, and is of size $w \times h$. We represent a cursor template T_i by a four-tuple (F_i, s_t, w_t, h_t), or its encapsulating frame, scale, width and height, respectively.

We make a first pass through the video and compute background difference images D_i for each frame F_i by using the mixture-of-gaussian (MOG) approach described in [34]. We apply morphological dilation and perform simple blob detection by discovering contours in the binary images D_i, and grouping cluster centers close to each other. A blob is defined as a region of connected white pixels in a binary image.

We denote the number of blobs obtained in D_i by $B(D_i)$ and define an indicator function for D_i, representing the occurrence of a single blob, given by:

$$1(D_i) = \begin{cases} 1, & \text{if } B(D_i) = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

A template T is added to a pool of potential templates $H(S)$, if a single blob is observed for N_T consecutive frames (Figure 3) starting at F_i and ending at F_{i+N_T-1}, or more formally

$$H(S) = \begin{cases} H(S) \cup T_{N_T}, & \text{if } \{\sum_{k=i}^{k=i+N_T} 1(D_k)\} = N_T, \\ H(S), & \text{otherwise.} \end{cases} \tag{2}$$

Initially, $H(S) = \varnothing$. Some cursor templates discovered in this step are shown in Figure 4. It is possible that some invalid cursor templates are discovered through this process. However, we observe that the number of valid cursor templates is far greater than the number of false positives. Furthermore, we may discover several similar cursors obtained in different temporal sections of the sequence. This increases the computational complexity of the template matching phase, as we are doing repeated work. As described in next section, we deal with this problem by selecting only the cursors discovered within the temporal vicinity of the frame being considered for template matching. This makes a trade-off between time complexity and tracking accuracy. Another way to reduce the number of duplicate cursor templates could be to use an algorithm such as k-means clustering to cluster similar templates together. Since we do not have prior knowledge about the number of cursor types present in the video, it is difficult to decide beforehand the number of clusters for the clustering algorithm. We thus use the entire pool of cursors $H(S)$ obtained in this step as candidate cursor templates for the template matching phase.

3.2. Template Matching

We make a second pass through the video, and each frame is pre-processed in two steps. The dilated background difference image computed in the previous phase, D_i, acts as a motion mask for the current frame F_i to remove background clutter, and the laplacian-of-gaussian filter is applied to this image to extract structural information. If no background motion is observed, i.e. $\sum_{j=0}^{m} \sum_{k=0}^{n} D_i(j,k) = 0$, the motion mask is not applied and the entire image is searched for the cursor template. We make an assumption that the target cursor in frame F_i can be determined from

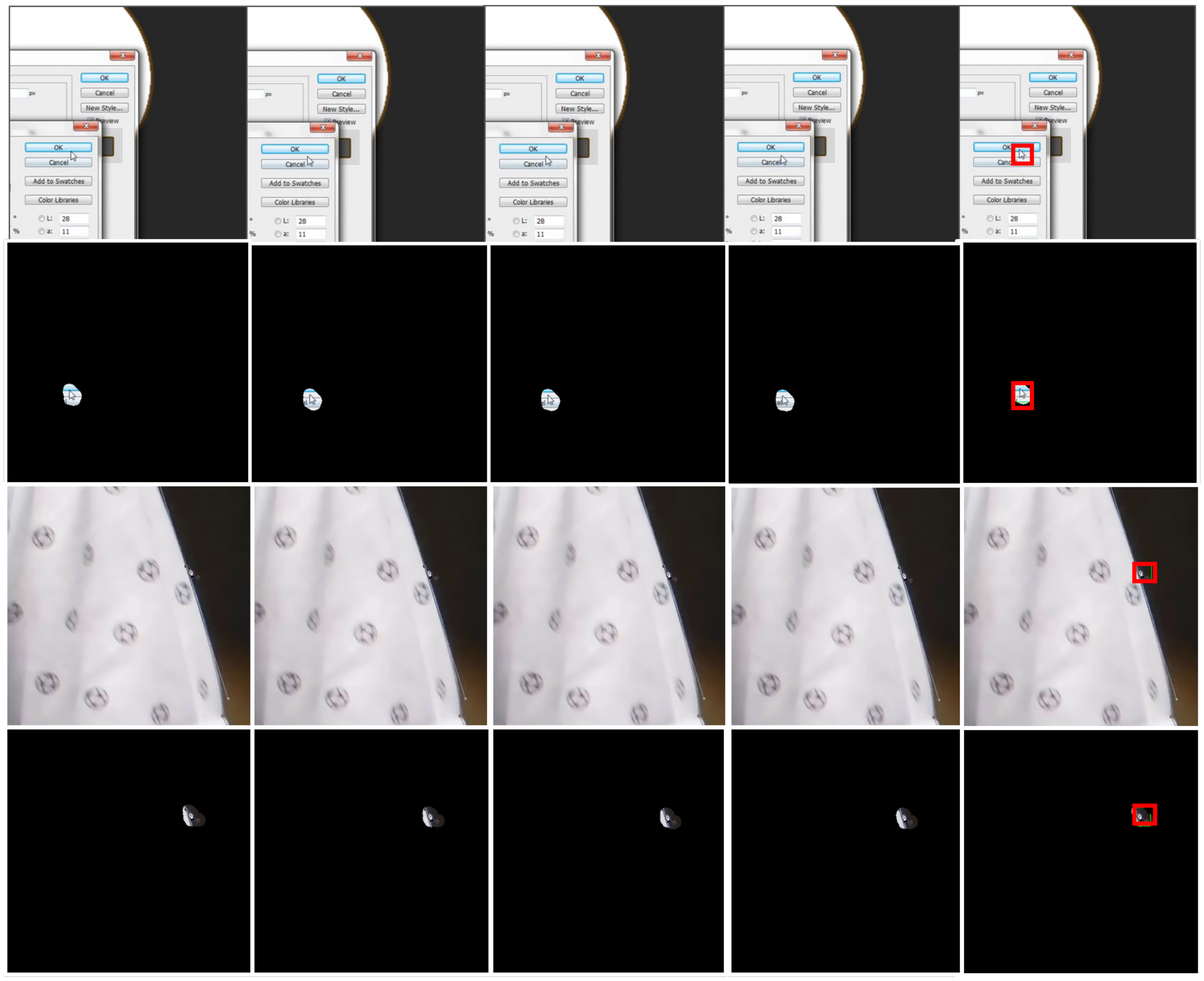

Figure 2. An illustration of the blob detection method used in the unsupervised cursor discovery stage. The blob present in the $N_T = 5$ frame (shown in red) is stored as a potential template in the pool of templates, $H(S)$ used in the next stage.

templates observed within a temporal vicinity, i.e. from frame F_{i-d} to F_{i+d} for some vicinity parameter d. We perform multi-scale template matching with template candidates discovered in frames $i-d$ through $i+d$, where each template is convolved across frame F_i to obtain a similarity map R where $R(x,y)$ stores the similarity score of a patch with its top-left location at (x,y). We use the normalized correlation coefficient for the similarity metric, as shown in Equation 3, where F denotes the frame, T denotes the template, and R denotes the resulting similarity map. The summation is done over $x' = 0...(w_t - 1)$, $y' = 0...(h_t - 1)$, where w_t and h_t are the width and height of the template and/or image patch respectively.

$$R(x,y) = \frac{\Sigma_{x',y'}(T(x',y').F(x+x',y+y'))}{\sqrt{\Sigma_{x',y'}T(x',y')^2.\Sigma_{x',y'}F(x+x',y+y')^2}} \tag{3}$$

We obtain a set of cursor proposals $u \in U_i$ (interchangeable with the bounding box representation defined earlier) for frame F_i as

$$U_i = \{u | u(x_{tl}, y_{tl}, x_{br}, y_{br}) \ \& \ M(u) >= 0.5\} \tag{4}$$

where each u is a cursor proposal represented by its top-left corner (x_{tl}, y_{tl}) and bottom-right corner (x_{br}, y_{br}) and is associated with a similarity score $M(u) = R(x_{tl}, y_{tl})$. Further, we apply non-maximum suppression to discard overlapping proposals in U_i, and retain the remaining proposals.

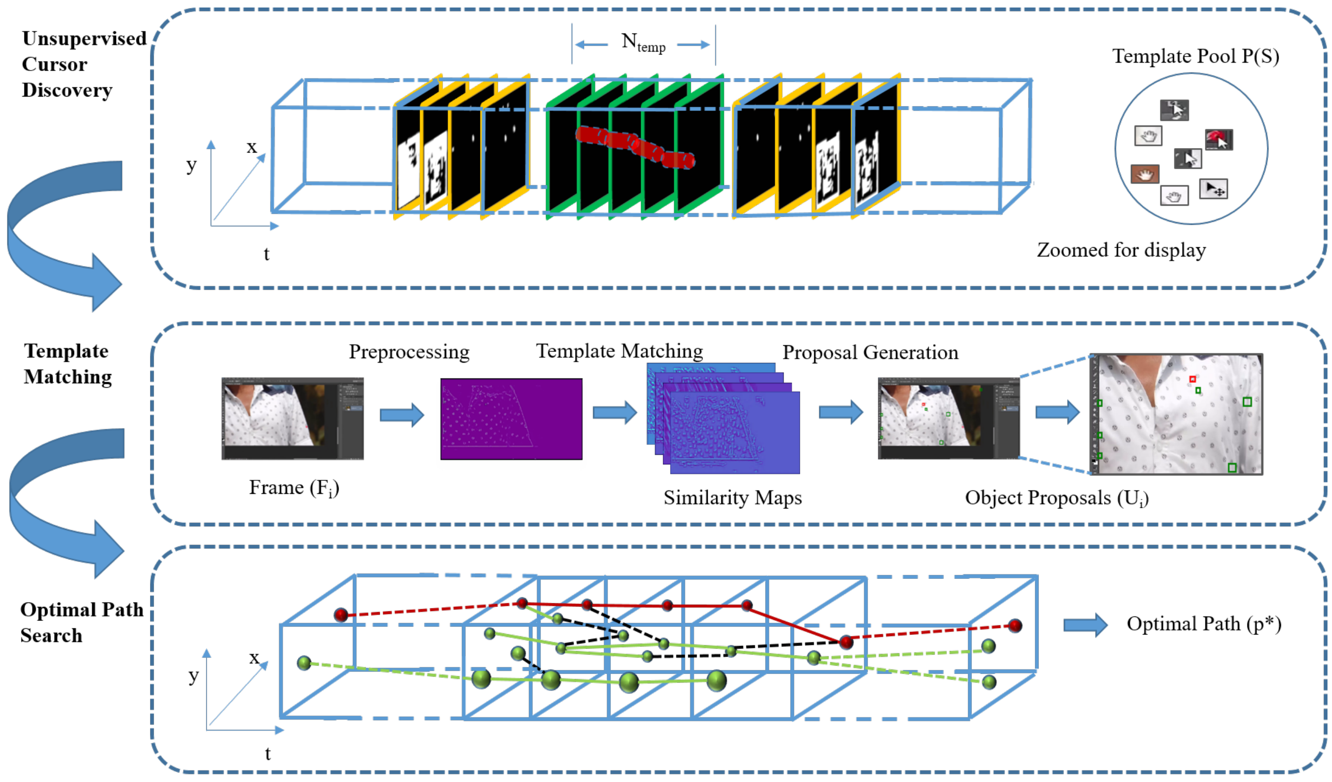

Figure 3. Our proposed method can be divided into three phases - cursor discovery, template matching and cursor tracking. We use blob detection to obtain a pool of templates P(S), perform template matching, and determine the optimal path in S, using the MaxPath algorithm.

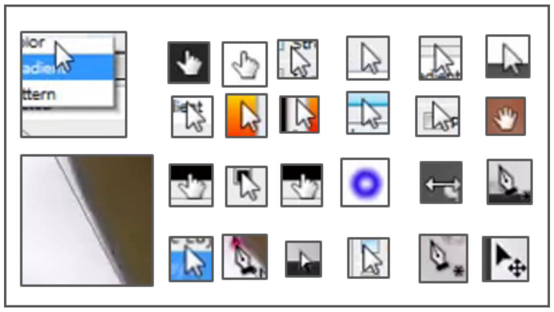

Figure 4. Examples of cursor templates obtained in the unsupervised cursor discovery stage.

3.3. Optimal Path Search

We perform mouse cursor detection by searching for an optimal spatiotemporal path in the video. A path p in a video, i.e. a frame sequence S, is defined by a sequence of cursor proposals $\{u_1, u_2, ...u_N\}$, one for each frame in S. The confidence score of a path, $M(p)$ is given by

$$M(p) = \sum_{i=1}^{N} M(u_i), \text{for } u_i \in U_i \tag{5}$$

We denote the set of all possible paths leading to the last frame F_N by P. Since there are U_N cursor proposals in F_N, the total number of possible paths leading to the final frame in the video sequence S is $|U_N|$. We are interested in locating an optimal path p^* with the maximum cumulative confidence score given by:

$$p^* =_{p \in P} M(p) \tag{6}$$

We use a variant of the MaxPath algorithm [32] to achieve this. Instead of maintaining a 3D trellis of spatiotemporal neighboring locations within a boundary region around u_i as proposed in [32], we link proposals in U_{i-1} to proposals in U_i if two path connectivity constraints are satisfied, 1) distance constraint - $E(u_{i-1}, u_i) <= t_{dist}$ where $E(u_{i-1}, u_i)$ indicates the Euclidean distance between the centers of object proposals u_{i-1} and u_i; and 2) scale constraint - $w_{u_{i-1}}/w_{u_i} \in [1-r, 1+r]$ where $0 < r < 1$, and w is the width of the cursor proposal. The distance constraint ensures that proposals in consecutive frames are not too far apart. The scale constraint enforces consecutive proposals to not drastically vary in size.

Video	Total Frames	Annotated Frames	Resolution
video1	546	388	1280 ×684
video2	500	446	1280 ×684
video3	1719	901	1212 ×720
video4	1988	359	1212 ×720
video5	485	483	1280 ×690
video6	491	277	1280 ×690
video7	484	234	1132 ×720
video8	510	507	1280 ×688

Table 1. Statistics of the dataset.

4. Experiments

4.1. Dataset

There are only a few published datasets to assist software intelligence algorithms, the most closely related to our problem being the PSOV dataset [7], a large dataset consisting of 74 hours of annotated video. However, their annotations are acquired for the purpose of command classification and recognition, and do not consist of mouse cursor bounding box information. Therefore, to evaluate our algorithm, we compile a dataset of 8 high-resolution videos, and compare the results with online tracking ([18], [1], [4]) and Faster-RCNN [29] detection baselines. We download the videos from YouTube [1], and obtain smaller sequences by selecting sections that can pose problems for object trackers. These sections include fast cursor movements; appearance shifts between different cursor types; zooms and pans of the editing window; and cursor disappearance and reappearance. Each video is of varying length, with the smallest video having 234 annotated frames, and the longest having 901 annotated frames, forming a total of 3595 annotated frames. The videos are manually annotated frame-by-frame using labelImg [2], by placing a bounding box around the mouse cursor (if present), and then assigning an appropriate cursor-type label to it. Since our baseline involves the training of a Faster-RCNN, we choose videos 1 to 5 as the training set for the Faster-RCNN and use videos 6 to 8 for test comparisons with our detector. Refer to table 1 for more details about the dataset.

We observed 17 different types of mouse cursors in the dataset, as shown in Figure 5. Certain cursor types (whitepointer, whitehand, nib) appear far more frequently than others (fade, stop, loading2) as seen in Figure 6, making it harder to train deep-learning-based object detectors such as the Faster-RCNN [29], which require a sufficient number of training examples for each cursor type.

[1]www.youtube.com
[2]https://github.com/tzutalin/labelImg

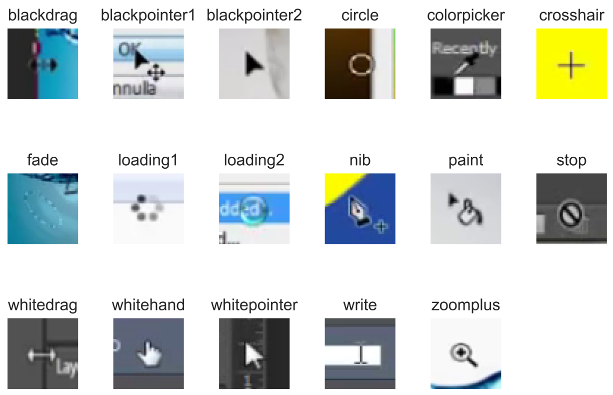

Figure 5. Cursor types present in the dataset.

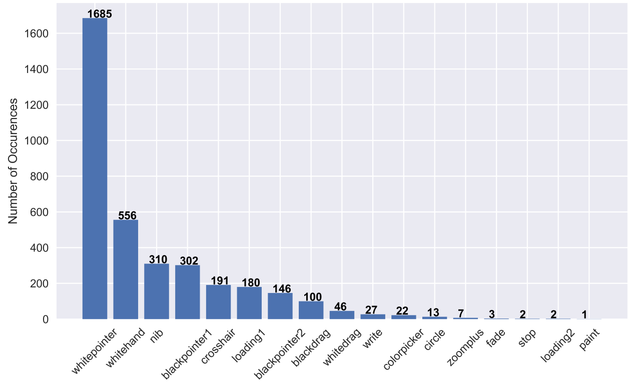

Figure 6. Distribution of cursor types present in the dataset.

4.2. Results

We compare our method with conventional online tracking methods - MIL [1], TLD [18] and Boosting [4]; and with Faster-RCNN [29], a deep-learning based object detection algorithm. We evaluate the methods using the video-intersection-over-union (V_{IOU}) metric which represents the overlap between the ground truth tube and the predicted tube in the video, and is defined as:

$$V_{IOU} = \frac{\sum_1^N R_t \cap R_{gt}}{\sum_1^N R_t \cup R_{gt}} \tag{7}$$

Here, R_t and R_{gt} represent the tracked and ground truth proposal region in each frame, respectively. $\cap$ and $\cup$ represent the intersection and union operators. The intersection-over-union (IOU) metric is a frame-level evaluation metric widely used in evaluating object detection frameworks and represents the degree of overlap between proposed and ground truth object bounding boxes. The V_{IOU} is a similar, but video-level metric.

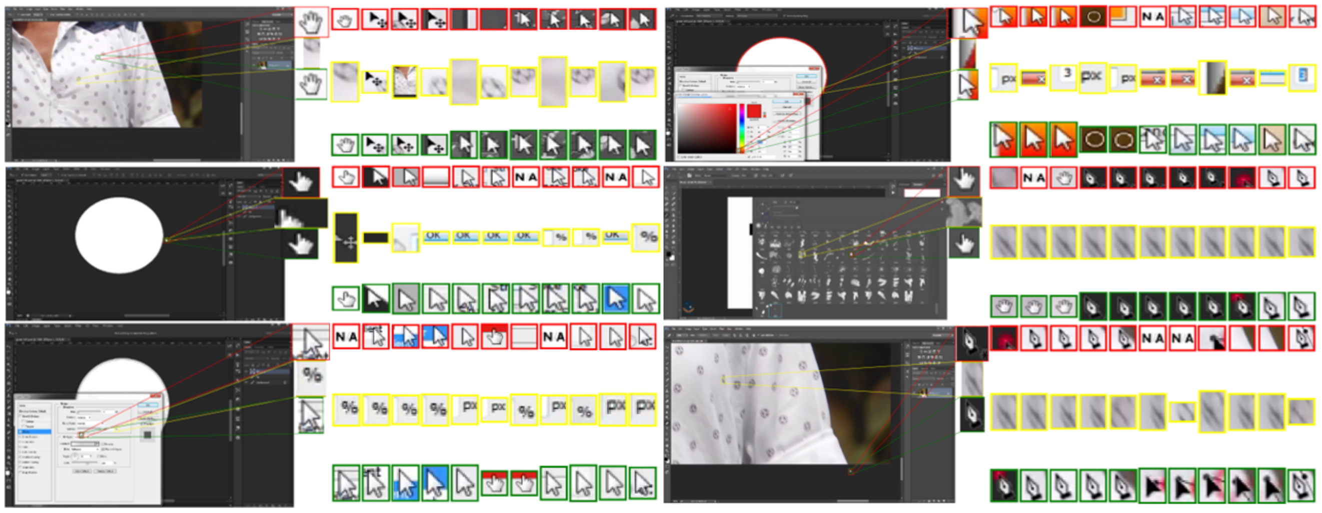
Figure 7. Qualitative comparison between our detector (red), Faster-RCNN (yellow) and ground truth (green) annotations. These results were obtained on test videos (video6, video7 and video8) from the dataset.

Method	V_{IOU}
Faster-RCNN (every 5th annotation, Train)	0.28 ($\pm$0.06)
Ours - Detection only (every 5th annotation, Train)	**0.327** ($\pm$**0.191**)
Faster-RCNN (every 5th annotation, Test)	0.05 ($\pm$0.02)
Ours - Detection only (every 5th annotation, Test)	**0.427** ($\pm$**0.112**)
Online Tracking (all annotated frames)	0.03 ($\pm$0.018)
Ours-Detection-All	0.355 ($\pm$0.161)
Ours-Tracking-by-Detection-All	**0.365** ($\pm$**0.146**)

Table 2. V_{IOU} Scores obtained using Faster-RCNN, online tracking, and our detection and tracking methods. Average V_{IOU} across all videos in each dataset partition are presented along with their respective standard deviations.

The online trackers are provided with the ground truth bounding box in the first frame, and are then applied to keep track of the target locations for the rest of the video. We only present the result produced by the best online-tracking method in Table 2. Although online tracking is not strictly comparable with our offline tracking-by-detection approach, we choose to present these comparisons to demonstrate the ineffectiveness of existing online tracking approaches for the cursor detection and tracking problem.

We use the Faster-RCNN implementation provided by Google's Object Detection API, with a ResNET-50 [15] base architecture, pre-trained on the MS-COCO dataset [21]; and then trained on frames from videos 1-5 from our dataset. We modify the default configurations to include smaller scales and aspect ratios in the anchor generation step; and use a binary classification objective, where the presence of a cursor, irrespective of type, is considered as a positive example. We evaluate the Faster-RCNN on every fifth annotated frame in each video. For a fair comparison, we present results using only the detection version of our approach, evaluated at every fifth frame, in which we discard the phase described in section 3.3, and choose the object proposal with the highest template matching score in each frame. We also present the detection and tracking-by-detection results evaluated on all 8 videos in the dataset using our approach. These are denoted by Ours-Detection-All and Ours-Tracking-by-Detection-All in Table 2. Our detection approach (evaluated on every fifth frame), achieves a V_{IOU} of 0.427 on the test set, thereby beating the Faster-RCNN counterpart which has a V_{IOU} of 0.067. Even the best online tracking method performs poorly on this dataset with a V_{IOU} of 0.03. It can also be seen that adding the tracking component (Ours-Tracking-by-Detection-All) results in an improved and less varying V_{IOU} score of 0.365 for the entire dataset, over pure detection (Ours-Detection-All), which has a V_{IOU} of 0.355 with a higher standard deviation. In Figure 7, we can observe better qualitative results from our detector as compared to Faster-RCNN, as our method is able to locate the mouse cursor when it undergoes instant appearance change, and in scenes with high background clutter. The Faster-RCNN detector is unable to locate the intended mouse cursor, and classifies similar looking patches in the background. This is because background clutter can be highly distracting due to occurrences of similar icons present in the background that bear resemblance to the intended mouse cursor, especially with no motion information to filter out unnecessary regions in the frame.

In the experiments, the consecutive frame requirement in Section 3.2, N_T is set to 5; the template selection vicinity, d and template similarity threshold t_temp in Section 3.3 were set to 150 frames and 0.5 respectively; and the distance and scale thresholds t_{dist} and t_{scale} in Section 3.4 were respectively set to 150 and 1.3. All experiments were conducted on a Windows 10 64 bit PC with Core i7-7700HQ @ 2.80 GHz CPU, 16 GB RAM and a 1060 Ti Nvidia GPU, with 6

GB memory.

5. DISCUSSIONS

In this paper, we proposed a novel offline, single-object tracking-by-detection algorithm for the purpose of automatically discovering and detecting mouse cursors in instructional videos. Our method implements tracking by detection, in three stages - automatic template discovery, template matching; and optimal path search - to adaptively discover and locate cursors appearing in a video. Experimental results show that our method performs much better than other online trackers, which are unable to track the cursor after a few initial frames. Our method is also able to locate cursors in an unsupervised manner, by using motion in each video to locate mouse cursors for template matching. Our pure detection approach outperforms the Faster-RCNN's performance on both the training and testing sets. On the other hand, Faster-RCNN is unable to generalize well to the test set, and this can be attributed to the small size of the training dataset, which is inadequate to train deep networks, despite the use of the pre-trained ResNet [15] architecture employed during the transfer learning phase. The poor performance of the Faster-RCNN can also be attributed to the significant differences between our dataset, and the MS-COCO dataset [21] which is used to pre-train the base ResNet. Our approach exploits the unique motion characteristics exhibited by the mouse cursor to discover potential cursor candidates appearing in a particular video. Template matching is a well-suited approach when we consider rigid objects with no pose variations, and is much faster during inference time when compared to Faster-RCNN. The use of optimal path search improves the tracking accuracy by filtering out unlikely paths with quick movements and drastic scale transformations, obtaining a spatiotemporal path with the highest confidence score.

References

[1] B. Babenko, M. Yang, and S. Belongie. Visual tracking with online multiple instance learning. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 983–990, June 2009. 2, 6

[2] J. R. Bergen, P. Anandan, K. J. Hanna, and R. Hingorani. Hierarchical model-based motion estimation. In *Proceedings of the Second European Conference on Computer Vision*, ECCV '92, pages 237–252, Berlin, Heidelberg, 1992. Springer-Verlag. 2

[3] L. Bertinetto, J. Valmadre, J. F. Henriques, A. Vedaldi, and P. H. S. Torr. Fully-convolutional siamese networks for object tracking. *CoRR*, abs/1606.09549, 2016. 2

[4] N. D. Binh. Online boosting-based object tracking. In *Proceedings of the 12th International Conference on Advances in Mobile Computing and Multimedia*, MoMM '14, pages 194–202, New York, NY, USA, 2014. ACM. 2, 6

[5] M. J. Black and A. D. Jepson. Eigentracking: Robust matching and tracking of articulated objects using a view-based representation. *International Journal of Computer Vision*, 26(1):63–84, Jan 1998. 2

[6] Z. Cai, Q. Fan, R. S. Feris, and N. Vasconcelos. A unified multi-scale deep convolutional neural network for fast object detection. *CoRR*, abs/1607.07155, 2016. 1, 2

[7] J. Cheng, H.-K. Hsu, C. Fang, H. Jin, S. Wang, and M.-H. Yang. Learning from photoshop operation videos: The psov dataset. In C. Jawahar, H. Li, G. Mori, and K. Schindler, editors, *Computer Vision – ACCV 2018*, pages 223–239, Cham, 2019. Springer International Publishing. 1, 6

[8] M. Cho, S. Kwak, C. Schmid, and J. Ponce. Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals. *CoRR*, abs/1501.06170, 2015. 3

[9] R. G. Cinbis, J. J. Verbeek, and C. Schmid. Multi-fold mil training for weakly supervised object localization. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2409–2416, 2014. 3

[10] L. Cui. MDSSD: multi-scale deconvolutional single shot detector for small objects. *CoRR*, abs/1805.07009, 2018. 1, 2

[11] T. Deselaers, B. Alexe, and V. Ferrari. Localizing objects while learning their appearance. In K. Daniilidis, P. Maragos, and N. Paragios, editors, *Computer Vision – ECCV 2010*, pages 452–466, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg. 3

[12] C. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg. DSSD : Deconvolutional single shot detector. *CoRR*, abs/1701.06659, 2017. 1, 2

[13] G. D. Hager and P. N. Belhumeur. Efficient region tracking with parametric models of geometry and illumination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(10):1025–1039, Oct 1998. 2

[14] K. He, G. Gkioxari, P. Dollár, and R. B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017. 1

[15] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 7, 8

[16] D. Held, S. Thrun, and S. Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference Computer Vision (ECCV)*, 2016. 2

[17] A. Joulin, K. Tang, and L. Fei-Fei. Efficient image and video co-localization with frank-wolfe algorithm. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 253–268, Cham, 2014. Springer International Publishing. 3

[18] Z. Kalal, K. Mikolajczyk, and J. Matas. Tracking-learning-detection. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(7):1409–1422, July 2012. 2, 6

[19] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'12, pages 1097–1105, USA, 2012. Curran Associates Inc. 2

[20] T. Lin, P. Dollár, R. B. Girshick, K. He, B. Hariharan, and S. J. Belongie. Feature pyramid networks for object detection. *CoRR*, abs/1612.03144, 2016. 1, 2

[21] T.-Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollr, and C. L. Zitnick. Microsoft coco: Common objects in context. *CoRR*, abs/1405.0312, 2014. 7, 8

[22] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. 1, 2

[23] B. D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, San Francisco, CA, USA, 1981. Morgan Kaufmann Publishers Inc. 2

[24] A. Lukei, T. Voj, L. ehovin, J. Matas, and M. Kristan. Discriminative correlation filter with channel and spatial reliability. *International Journal of Computer Vision*, 126, 11 2016. 2

[25] L. Matthews, T. Ishikawa, and S. Baker. C7. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(6):810–815, June 2004. 2

[26] H. Nam and B. Han. Learning multi-domain convolutional neural networks for visual tracking. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016. 2

[27] A. Prest, C. Leistner, J. Civera, C. Schmid, and V. Ferrari. Learning object class detectors from weakly annotated video. In *CVPR*, pages 3282–3289. IEEE Computer Society, 2012. 3

[28] J. Redmon, S. K. Divvala, R. B. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015. 1

[29] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 91–99, Cambridge, MA, USA, 2015. MIT Press. 1, 6

[30] Z. Shi, T. M. Hospedales, and T. Xiang. Bayesian joint topic modelling for weakly supervised object localisation. *CoRR*, abs/1705.03372, 2017. 3

[31] K. D. Tang, A. Joulin, L.-J. Li, and L. Fei-Fei. Co-localization in real-world images. *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1464–1471, 2014. 3

[32] D. Tran, J. Yuan, and D. A. Forsyth. Video event detection: From subvolume localization to spatiotemporal path search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 36(2):404–416, 2014. 1, 5

[33] L. Wang, G. Hua, R. Sukthankar, J. Xue, and N. Zheng. Video object discovery and co-segmentation with extremely weak supervision. 09 2014. 3

[34] Z. Zivkovic and F. van der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern Recognition Letters*, 27:773–780, 2006. 3