Crowd-Guided Ensembles: How Can We Choreograph Crowd Workers for Video Segmentation?

Alexandre Kaspar [1] **Geneviève Patterson** [2] **Changil Kim** [1]
Yağız Aksoy [1,3] **Wojciech Matusik** [1] **Mohamed Elgharib** [4]
[1] MIT CSAIL [2] Microsoft Research NE, [3] ETH Zürich, [4] HBKU QCRI
{akaspar, changil, yagiz, wojciech}@mit.edu, gen@microsoft.com, melgharib@hbku.edu.qa

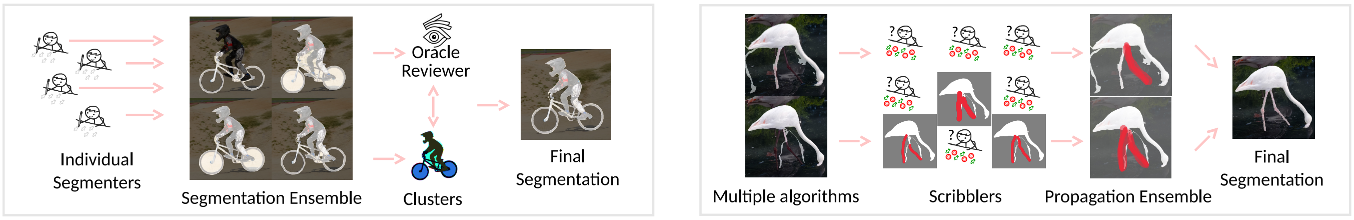


Figure 1. An illustration of our two proposed crowd-guided ensemble methods. *Left:* Our segmentation ensemble combines the results of multiple crowd workers through the guidance of an oracle reviewer. *Right:* Our propagation ensemble gathers the information about where multiple distinct algorithms fail from the accumulated scribbles of crowd workers and merges it into the result that incorporates the best of each algorithm.

ABSTRACT

In this work, we propose two ensemble methods leveraging a crowd workforce to improve video annotation, with a focus on video object segmentation. Their shared principle is that while individual candidate results may likely be insufficient, they often complement each other so that they can be combined into something better than any of the individual results—the very spirit of collaborative working. For one, we extend a standard polygon-drawing interface to allow workers to annotate negative space, and combine the work of multiple workers instead of relying on a single best one as commonly done in crowdsourced image segmentation. For the other, we present a method to combine multiple automatic propagation algorithms with the help of the crowd. Such combination requires an understanding of where the algorithms fail, which we gather using a novel coarse scribble video annotation task. We evaluate our ensemble methods, discuss our design choices for them, and make our web-based crowdsourcing tools and results publicly available.

ACM Classification Keywords

H.5.3. Group and Organization Interfaces: Computer-supported cooperative work; I.4.6 Image Processing and Computer Vision: Segmentation

Author Keywords

crowdsourcing; video object segmentation; ensemble methods; keyframe segmentation; segmentation propagation

CHI 2018, April 21–26, 2018, Montreal, QC, Canada
© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ISBN 978-1-4503-5620-6/18/04...$15.00
DOI: https://doi.org/10.1145/3173574.3173685

INTRODUCTION

Video segmentation is one of the most essential tools for movie post-production and more recently for generating training data for a multitude of data-driven algorithms. The current practice heavily depends on specialized rotoscoping artists who utilize several commercial software products, often in orchestration. The dependence on specialized artists results in an excessive financial cost and makes rotoscoping less accessible. In this paper, we aim to democratize rotoscoping by simplifying the work of the artist into a less intensive, reviewing role that supervises a distributed crowd workforce.

Crowdsourcing is a widely used tool for distributing large manual tasks to a group of inexperienced workers. In visual data processing, it is widely used for 2D image-space operations like image segmentation. In the previous efforts for crowdsourced image segmentation [6, 7, 31, 45], the segmentation results are accepted from a single worker's result in its entirety, discarding the efforts of the rest. This approach wastes the full potential of the crowd since it only keeps the result of a single best worker. Moreover, these methods do not trivially extend to video segmentation as it requires a careful treatment of temporal coherency, and a frame-by-frame application on multiple frames may result in a prohibitive cost and thus not be scalable. The endeavor to propagate image segmentation to videos is not mature enough and a general solution to this problem is under active research [38].

We propose a novel crowdsourced video segmentation workflow that allows and deliberately utilizes the redundancy of input, from both crowd workers and automated algorithms. We make use of two novel *ensemble methods* in two key steps frequently arising in modern video segmentation pipelines [1, 10], namely keyframe segmentation and propagation (see Figure 1). Our focus is on exploring new collective capabilities of the crowd in complex macrotasks where skills are needed, and ambiguities are common. We primarily target novel crowd interactions with automatic methods, while following a desire for higher segmentation *quality* and *scalability*. Our main contributions consist of:

- The acquisition of higher-quality segmentation by merging the work of multiple crowd workers,
- The improvement of crowdsourced segmentation capabilities with the introduction of *negative polygon annotations*,
- A novel coarse scribbling task to merge multiple automated propagations locally with the guidance of the crowd,
- The application of our scribble annotations to delegating segmentation review to the crowd, and further to propagating trimaps for video matting, illustrating their flexibility.

We evaluate our methods and design decisions thoroughly and discuss their merits and shortcomings. We make our tools and annotation results public to facilitate future research: see http://crowdensembles.csail.mit.edu.

RELATED WORK

A canonical system for human-in-the-loop video object segmentation is rotoscoping [10]. An input video is isolated to cuts, which are then decomposed into keyframes. The keyframes are manually segmented and then propagated to neighboring frames automatically using motion cues and image features. Finally, the propagated segmentations are refined manually [1]. This process is repeated until the desired quality is achieved. Since it requires skilled artists and its pace is rather slow (fifteen frames per day in average [30]), the rotoscoping pipeline is cost-intensive, seldom scalable and poorly accessible. Nonetheless, it is a crucial tool in diverse areas from simple image composition to visual effects used for movie productions to the generation of ground truth annotation on which today's artificial intelligence (AI) engines depend. This work tries to make this process more accessible by relying on a crowd workforce.

Crowdsourced Image Segmentation

Recent crowdsourcing systems for image segmentation include OpenSurfaces [7], upon which our segmentation interface is based; Intrinsic Images in the Wild [6]; Materials in Context [8]; the hierarchical instance segmentation pipeline of Microsoft COCO [31]; and the Video Annotation Tool from Irvine, California [45]. They share a common strategy: validating a single worker's annotations. Our approach embraces the small, local errors that different annotators or automatic methods make and relies on workers to combine multiple segmentations into a higher-quality result.

Hybrid Crowd-AI Systems

Many crowd-annotation systems improve on challenges such as scalability and cost-effectiveness [16, 35, 44], worker consensus [20, 18], annotation quality [21], crowd-AI interaction algorithms [9, 36, 28], identifying breakage in automated labeling [48], and workflow control [14]. In their paper on the future challenges facing crowd work, Kittur et al. [27] emphasize that crowd-guided AI systems are a core problem of this field. Our work shares the same intent of recent crowd-AI systems [22, 29, 19] to combine the knowledge of the crowd with machine learning and computer vision, and similarly treats the crowd as a core component of the full system. We present the first attempt to have ensembles of crowd workers guide automatic video segmentation.

Video Segmentation Propagation

Segmentation propagation plays a crucial role in reducing the workload of per-frame segmentations and is extensively studied in the video segmentation literature [13, 37, 33, 42]. Most recent offline approaches are increasingly using complex data-driven models that rely on the availability of video segmentation datasets, which the recent benchmark of Perazzi et al. [38] points out the scarcity of, as well as their limited size and variations. On the other side, interactive methods [5, 39, 47, 32, 41] rely on complex hand-crafted features and cues. This work leverages the diversity of propagation methods combined with a crowd workforce to implicitly select the best features and propagation strategies.

ENSEMBLE METHODS USING THE CROWD

Ensemble methods combine multiple results to achieve higher quality than any of the individual results [17]. In crowdsourcing systems, it is common practice to ensemble the results of tasks which exhibit high variance inherent to the varying degrees of human skills, attention or complex internal motivations. The most common approach is to make a Bayesian decision, which defaults to a simple voting scheme with votes possibly weighted by their confidence if such information is available.

Most crowd-AI systems eventually make use of an ensemble strategy. However, these primarily happen in microtasks where the result can be merged automatically or easily evaluated by a human. Image and video segmentations are macrotasks for which ensembles have not yet been proposed. Furthermore, their quality evaluation is complicated because they contain many ambiguities such as motion blur, light interactions, clutter and other visual artifacts. Our results show that segmentations are often neither good nor bad as a whole.

We propose the use of novel ensemble strategies in two components of our video segmentation pipeline: first for *individual frame segmentations* and second in merging multiple corrections from the crowd in *propagated segmentation* results. We do so with the considerations to generate high-quality video segmentation results from the crowd.

Segmentation Ensembles

The standard approach for single-frame segmentation acquisition in large crowdsourcing efforts such as OpenSurfaces [7] and COCO [31] consists of using a secondary task to evaluate the quality of each individual segmentation. If any single segmentation is evaluated as sufficient, it is accepted, else it is rejected and a new segmentation is requested. With a known set of expert segmenters [34] or qualified workers [7], the evaluation task can be sidestepped and the segmentation is directly accepted. While this approach results in a small cost per segmentation, it may not be sufficient for high-quality results using the crowd because no single worker may ever reach the desired quality. Furthermore, the evaluation strategy

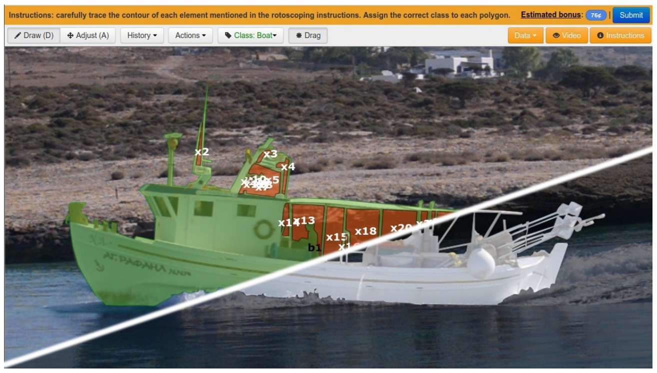

Instructions: carefully trace the contour of each element mentioned in the rotoscoping instructions. Assign the correct class to each polygon. Estimated bonus: 76¢ | Submit

Figure 2. Our segmentation user interface with positive and negative polygons for the *boat* sequence (*top-left*) and the corresponding segmentation overlaid (*bottom-right*).

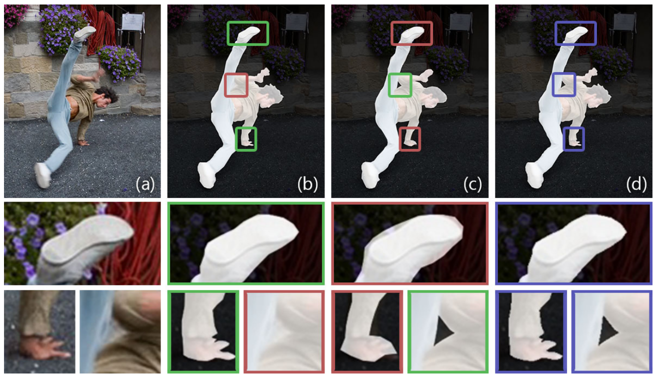


Figure 3. For an input image (a), we merge multiple segmentations, such as (b) and (c), so that the end result (d) is able to keep the best regions of them (shown in *green insets*) and reject other regions where they underperform (shown in *red insets*).

assumes some quality threshold decided by the crowd which is highly subjective as noted in OpenSurfaces [7] and may not necessarily match the desired quality.

Instead of using pass/fail evaluations of individual segmentations, we propose a system that is designed to take advantage of multiple distinct segmentations to achieve a high-quality final result. Furthermore, we introduce the use of *negative annotations*, which makes it easier for the segmenters to generate detailed results such as the one shown in Figure 2.

Negative Space Annotation

We developed a polygon-based segmentation interface based on OpenSurfaces [7] with one major modification: the crowd workers have the option to use *negative* polygons that subtract regions from the foreground in addition to the standard, positive polygons. The idea is motivated by the extensive use of *negative space* in rotoscoping [10]. In practice, this allows higher-quality segmentations as real, complex objects often contain small regions where the background is visible as illustrated in Figure 2. Defining complex polygons with negative annotations introduces a process ambiguity: one can segment only the positive space by decomposing it into multiple components, or one can segment the whole target as one positive region that possibly includes several negative components. We solve this ambiguity in our merging stage and do not dictate which strategy to use as both have advantages and disadvantages.

Merging Segmentations

Given N segmentations of the same object by distinct segmenters, we obtain our segmentation result in our review phase. We assume the existence of an oracle, who can be a single user or the crowd, who provides us with weights w_i that represent the relative qualities of each segmentation $i = 1, \dots, N$.

When dealing with complex segmentations made of multiple polygons, treating the whole segmentation as a single merging operation breaks polygons that have no counterpart in the other segmentations. Thus, we first cluster polygons into minimal disjoint clusters such that no polygon intersects a polygon from another cluster. Furthermore, we treat each polygon class (positive or negative) as a separate layer (foreground or background) and merge polygons within their cluster and class separately. This is especially important to make full use of the negative polygons, which our experiments showed to be an indication of higher quality. Given a specific class layer and cluster C, we take per-worker sub-segmentations made of their corresponding polygons, and merge them pixel-wise by weighted average voting:

$$\mathrm{M} = \left[\frac{1}{W} \sum_{i=1}^{N} w_i \mathrm{M}_i \right]_{\geq 0.5}, \qquad (1)$$

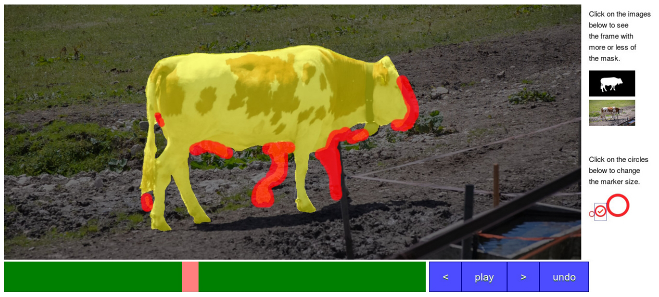
where $[\cdot]$ denotes a thresholding operator resulting in a binary value in $\{0, 1\}$, $\mathrm{M}_i \in C$ a sub-segmentation of worker i, and $W = \sum_i w_i$ the total weight. Finally, we take the union of the non-overlapping clusters that were generated by a weighted majority of workers, and merge the foreground layer F with the background layer H into the full segmentation $F \cap \neg H$.

While our strategy could be suboptimal in isolated cases, it minimizes the work of the reviewer and already achieves a significant overall quality improvement over using only one of the input segmentations, or using a simple voting strategy. An example merging result can be seen in Figure 3.

Oracle Review

In our implementation, the oracle selecting the individual weights w_i is the requester (or reviewer) who accepts or rejects results and distributes the money to the crowd workers. The review process reduces to choosing weights for each of the N results. Selecting the weights $w_i = 1 + \beta_i$ serves two purposes: (i) to select the best result composition with weights w_i, and (ii) to supplement the base task reward with additional financial bonuses β_i (in cents).

When crowdsourcing large segmentation acquisitions, an extra problem arises with the reward selection. Online platforms such as Amazon Mechanical Turk require the selection of a reward to publish a task. This requirement implies that the requester must evaluate the complexity of the task to choose an appropriate reward. Instead of relying on some complexity assessment, we use a low base reward, which is then complemented with financial bonuses β_i directly derived from the merging weights chosen by the oracle during the review. The base reward encapsulates both the maximal amount of money we are giving for any segmentation and the minimum reward


Click on the images below to see the frame with more or less of the mask.

Click on the circles below to change the marker size.

Figure 4. Our user interface for crowd workers used in the scribble task for the propagation ensemble. The propagated segmentation is shown in *yellow*, while user scribbles are marked *red*.

that complex segmentations require to attract enough good segmenters.

Propagation Ensembles

Given the keyframe segmentations, a standard rotoscoping pipeline attempts to propagate them to the neighboring frames. We assume that we have a number of algorithms available to do this and that they each make different assumptions so that there are variances among their propagation results.

As it is unlikely for any single segmentation propagation method to produce satisfactory results for all frames in all scenarios, we define a new segmentation merging strategy. The most straightforward way to merge them is by per-pixel majority voting. This, however, does not include any knowledge of where different methods likely fail and is only effective when the majority of the methods perform well. Instead, our strategy relies on weak annotations from the crowd that describe where the different methods fail. These annotations need not be precise and thus take less time than full segmentations.

Scribble Annotation

Our interface for this task includes a slider bar to go over the frames of the video interval and two sets of buttons: one to vary the opacity of the overlaid segmentation result on the input frame, and the other to set the size of the scribble brush as shown in Figure 4.

For each of K different propagations given for a frame, we first merge the crowd scribbles to create *scribble heat maps* S^k:

$$\mathrm{S}^k = \frac{1}{B} \sum_{i=1}^{N} b_i^k \, \mathrm{S}_i^k \,, \qquad (2)$$

where $b_i^k \in [0.25, 1]$ is a weight inversely proportional to the *average brush size* used by the worker i for the propagation algorithm k, S_i^k is the corresponding scribble map, and $B = \sum_i b_i^k$. The average brush size for the scribble S_i^k is computed as the average of stroke brush sizes weighted by their corresponding stroke length.

We then merge the scribble heat maps and the propagation results P^k to get our segmentation result:

$$\mathrm{M} = \left[\sum_{k=1}^{K} \mathrm{P}^k \circ \mathrm{S}^k \right]_{\geq A} \,, \qquad (3)$$

where $\circ$ is a pixelwise operator and A is a pixelwise threshold. We investigated three different interpretations of our scribble task, each of which corresponds to a different merging operation $\circ$. Each of these has pros and cons which we detail in the experiment evaluations.

Scribble as Error Correction

We use the scribbles to invert the erroneous regions of segmentation propagations. The merging operator $\circ$ is defined as

$$p^k \circ q^k = \begin{cases} p^k - q^k & \text{if } p^k = 1 \\ q^k & \text{if } p^k = 0 \end{cases} \,, \qquad \text{(pxor)}$$

where $p^k \in \{0, 1\}$ denotes a pixel of the binary mask P^k, $q^k \in [0, 1]$ is from the scribble map S^k, and the threshold $A = K/2$, i.e. the inversion only occurs if the majority requires it.

Scribble as Soft Penalty

We use the scribbles to locally penalize a method. This can be interpreted as annotating the regions where we do not trust the segmentation propagation. The merging operator $\circ$ is here defined as

$$p^k \circ q^k = p^k (1 - q^k)^{\alpha} \,, \qquad \text{(wmaj)}$$

where we use the per-pixel threshold $A = [\sum_k (1 - q^k)]^{\alpha}/2$ and power $\alpha = 2$ for smoothness.

Scribble as Segmentation Refinement

We use the scribbles to locally overwrite the segmentation. In this scenario, we add a brush type selection. Users can use two different brushes to either scribble foreground or background. When a new brush stroke overlaps with an old one, the overlap region is replaced so that every pixel of a worker scribble is either *positive* (foreground), *negative* (background) or *undefined* (no scribble). In this case, merging consists of generating two different scribble heat maps S_+^k and S_-^k for the foreground respectively the background, and using the following $\circ$ merging operator

$$p^k \circ (q_+^k, q_-^k) = \begin{cases} 1 & \text{if } q_+^k = 1 \\ 0 & \text{if } q_-^k = 1 \\ p^k & \text{otherwise} \end{cases} \,, \qquad \text{(2-brushes)}$$

where q_+^k and q_-^k are from the corresponding scribble maps S_+^k and S_-^k and the threshold is $A = K/2$.

Figure 5 illustrates a positive example of using either soft penalty or error correction to merge multiple propagations so that the result is better than any of the original segmentation propagations.

Brush Size Regularization

In our interface, we provide three different brush sizes corresponding to radii of $b \in \{8, 16, 32\}$ screen pixels. When processing the data, the natural normalization consists of scaling the brush sizes to its equivalent in image space, i.e.

$$b_{\text{image}} = \alpha \times b_{\text{screen}}$$

where α is the ratio of the image width in image space to its width in screen space. In our setup, we typically had the fixed ratio $\alpha = 1920/800 = 2.4$. We consider normalization by a

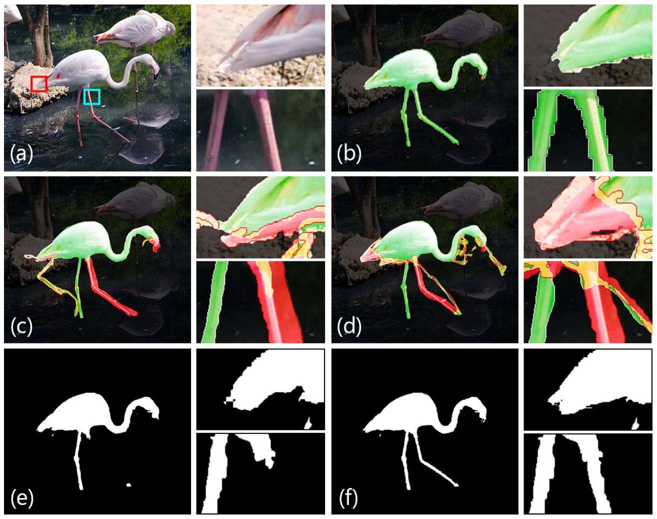


Figure 5. Example propagations using three different methods: (b) The first one is more robust but the result is not as smooth. (c, d) The two methods produce large artifacts, but the results are in general very smooth; Green overlays represent the propagated segmentations, red the scribble heat map, and yellow their intersections. The bottom row shows majority merging (e), and the proposed scribble merging (f) using either corrective or penalty-based merging as their results are similar.

scaled version of α, i.e. $\alpha' = f \times \alpha$ where $f \neq 1$ corresponds to using an image space brush that is either smaller or larger than what was expected by the user to normalize their input. In our experiments normalizing to a smaller effective brush size ($f < 1$) leads to better results.

Scribble Outliers

Given the nature of the task, it would be hard to have a meaningful review process for the results. Thus we chose to accept all results but those that are empty. This means that we have to deal with potential outliers in our scribble results. Our solution is to use a modified constant K when thresholding. Instead of using the number of actual workers for the given frame, we replace it with

$$K' = \max_{p \in S} \sum_{k=1}^{K} p^k , \tag{4}$$

i.e. we use the maximum overlap of all workers as effective number of valid workers.

EXPERIMENTS

We evaluate our two proposed ensemble methods in the context of a crowdsourced pipeline that follows the conventional keyframe segmentation and propagation strategy of rotoscoping. The evaluation is done using the DAVIS dataset [38] and its three metrics: *region similarity* J, *contour accuracy* F, and *temporal stability* T. The first two metrics J and F respectively measure the amount of correct pixelwise overlap of segmentations (commonly referred to as intersection over union), and the quality of the segmentation boundaries. The values are each in the interval of $[0,1]$; the higher the better. The last metric T measures the temporal smoothness, for which lower values correspond to better temporal transitions of segmentations. The dataset consists of 50 short video sequences that have been annotated by a rotoscoping artist, for a total of 3455 frames at 1080p HD resolution.

The propagation part of the evaluation is based on our crowdsourced segmentation results. This differs from the typical evaluation of semi-supervised and unsupervised methods on DAVIS in that we cannot assume we have access to the ground truth since our goal is to have the crowd generate it. Thus, for all our automated propagations, the training data we use comes from the results of our crowd workers and not from the original DAVIS dataset.

All of our experiments were done on the Amazon Mechanical Turk platform. For all the evaluation figures including those related to scribbles, we used the *pxor* merging method with brush regularization $f = 1/2$ unless stated otherwise.

We refer the readers to the supplementary material accompanying our paper for screenshots, videos, and the code demonstrating example sessions using our crowdsourcing user interface.

Segmentation Experiments

Our baseline segmentation is acquired by sending each segmentation to 3 different workers. We initially allowed workers with a global success rate higher than 50% to work on our segmentation tasks, and refined our worker group later using a whitelisting strategy, where workers were assigned a custom accreditation maintained by us. Our segmentation results have been generated by a set of 70 best-performing workers.

The instructions of this task contain both (1) generic segmentation instructions including examples of good and bad segmentations unrelated to the current task, and (2) a sequence-specific set of instructions that explains what exactly should be segmented for the current task. This includes sentences describing the target as well as an example of a good segmentation. Regarding the usage of *negative* polygons, workers are shown a preview of the generated segmentation to validate their work at submission time.

The results are then reviewed by the main user, or *the oracle* as mentioned before, and merged according to their chosen weights. The review is done with an interface that displays a variety of information including the original image, the actual segmentation masks, their color-coded differences as well as the current merged result for the interactively chosen weights. To enable fair rewards beyond the low base reward of $0.15, the interface also measures the time taken by a crowd worker to do the segmentation, based on the timestamp of each action, and provides correspondingly the lack, or excess, of reward given the current weights, which directly translates into bonuses.

To measure the impact of replication, we picked the fifth and fifth-to-last frames of each sequence, and send them to be segmented seven more times (thus totaling ten times). We repeat this once using whitelisting and once not. No review was done for these extra segmentations.

Lastly, we experimented on automating the review process of our oracle user by sending a scribble task to annotate the worker segmentations. The task required scribbles for each of the three initial segmentation of the fifth and fifth-to-last frames of each sequence. We then merge the scribbles with the individual segmentations to get a scribble-based review.

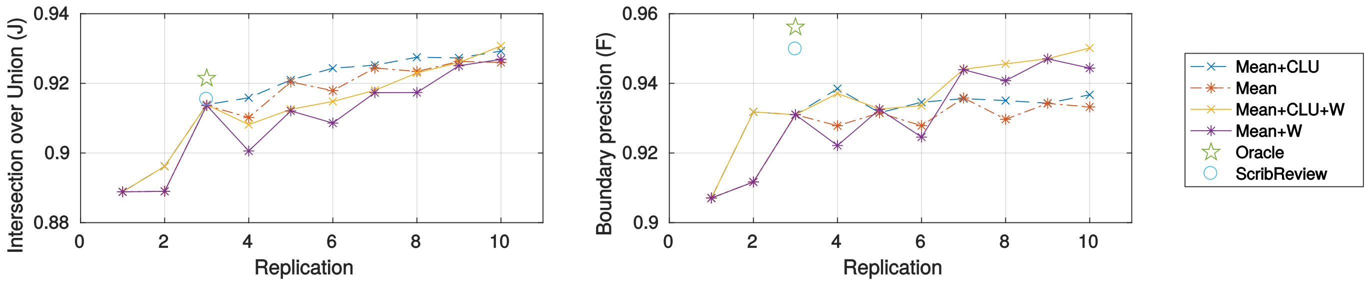


Figure 6. Evolution of the quality with the increasing segmentation replication for two automatic merging strategies: Mean+CLU is our strategy, whereas Mean indicates the simplistic pixelwise majority. +W represents the variants where the extra seven replications were acquired through the whitelisted group (thus totaling ten replications). The default extra replications were acquired without accredition requirements. Clustering seems to always be a better solution, especially for even replication counts, where ties happen. Whitelisting seems to bring a mixed consequence: in our setup, it did not improve the coverage (J) except for high replication counts, but it did substantially improve the boundary quality (F). The scribble-based review produces the results that are significant better in terms of the boundary accuracy but not so much for the coverage. However, tt does not reach the same quality of the oracle reviewer.

Note that we do not distinguish this task from the ordinary reviewing task: workers see the three segmentations with the same instructions as for other scribble tasks.

Segmentation Results

We evaluate our proposed ensemble method for keyframe segmentation, according to the following three criteria: (1) the usage and efficacy of our novel negative space annotation tool; (2) the efficacy of merging multiple segmentations in terms of the quality of the final result, with varying merging strategies and replication counts; and (3) the possibility of delegating the role of the oracle user to crowd workers.

Use of Negative Space Annotation

The concept of negative space is standard in composition in the visual arts. However, to the best of our knowledge, it is the first time to be used in a crowdsourced interface for high-quality image segmentation using *negative polygons*, which then raises the question: can workers make good use of it?

With all worker segmentations of all sequences counted, crowd workers created 34,761 polygons, of which 19,985 (57%) were negative polygons. The number and percentage of negative polygons vary significantly with sequences. For instance, the simple *blackswan* sequence of DAVIS required 8 negative polygons out of 172 polygons (4%), whereas the more complex *boat* sequence consisted mostly of negative polygons (4006 of total 4409 polygons, amounting to over 90%; see Figure 2 for an example segmentation). The presence of negative annotations in a segmentation weakly correlates with its J value being above average for that segmentation ($r = 0.17, p < 0.01$) and similarly with its F value ($r = 0.26, p < 0.01$).

Segmentation Quality

In Table 1, we provide the results for the naive approach of fully segmenting the video sequences through keyframe segmentation tasks. Two major observations are: (1) the naive full segmentation does not ensure any temporal coherence—our workers perform independent segmentations—and thus a high performance in the temporal stability metric T is not expected; and (2) segmentations done by workers do not always yield similar level of segmentation quality around fine details such as thin occluders and intricate boundaries.

	Oracle	Automatic			
Metrics	**Clu/Mean**	Best	Clu/Mean	SDF	Worst
$J \uparrow$	**0.917**	**0.914**	0.903	0.875	0.842
$F \uparrow$	**0.952**	**0.939**	0.928	0.880	0.879
$T \downarrow$	0.380	0.372	0.359	**0.350**	0.483

Table 1. Comparisons of different merging strategies for the naive ground truth acquisition using the DAVIS dataset [38]. Clu/Mean: both the cluster-based pixelwise majority and the pure pixelwise majority; Best: the single best-performing segmentation of the batch; SDF: the average signed distances from the segmentation boundaries; and Worst: the single worst-performing segmentation. The metrics J, F and T are defined in the text. For these results, the replication count is set such that the clustering produces similar results to the direct pixelwise majority.

The table further shows the effect of our merging procedure for the segmentation task when compared to alternative strategies. This justifies our use of multiple results instead of a single one, and also shows that our merging strategy performs better than the two obvious alternatives: using a uniform average, i.e. the same weight for every worker's result; or using the single best result, i.e. reject all but the best. Note that our method reduces gracefully to the second alternative in the presence of an outstanding worker.

Impact of Replication

Figure 6 shows the evolution of the segmentation quality as we increase the replication count from $R = 1$ to 10. Odd counts tend to be better since they avoid ties during majority voting. Our clustered merging strategy seems to be always better than the default pixelwise merging and does especially better for even replication counts. However, clustering does not seem to improve quality substantially for odd replication counts. The whitelisting strategy did not improve coverage (J), but did improve boundary accuracy (F). This suggests that replication helps with coverage, but higher skills are needed for finer details.

Delegating the Review to the Crowd

Figure 6 also shows the result of using crowd scribbles instead of our oracle-based review. It does not seem to produce a significant coverage (J) improvement to the default pixelwise averaging, but it does significantly improve the boundary accuracy (F). While the crowd does not reach the level of a dedicated oracle user, our results show that it is a viable al-

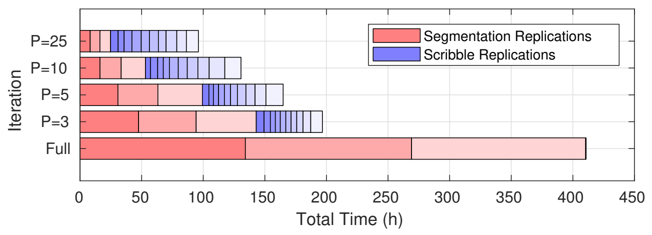


Figure 7. Cumulative timings of the segmentation and scribble replications for sampling intervals $P = 25, 10, 5, 3$ and the full segmentation.

Task	Segmentation	Scribble
Frame cost	$ 0.15	$ 0.015
Bonus (on average)	$ 0.15	-
Replication	$\times 3$	$\times 10$
Total cost (on average)	$ 0.90	$ 0.15

Table 2. Per-frame, per-worker cost breakdown. Note that a scribble task consists of multiple frames and methods. Here we report only the effective cost per frame. The bonus is taken as an average.

ternative if the pipeline is desired to be more automated and scalable through crowdsourcing.

Propagation Experiments

In order to evaluate our propagation ensemble method, we make use of two complementary classes of propagation methods. Note that our method is agnostic to which propagation algorithm is used and the choices presented below can easily be replaced with any future propagation algorithm. In our experiments, the keyframe segmentations are propagated to the others using two classes of algorithms detailed below.

Optical flow–based propagation: we warp a given segmentation at frame t to frames $t \pm k$ using the optical flow computed between them with large-displacement optical flow [11]. The flow is computed forward and backward, resulting in two distinct propagations.

Feature–based classification: to complement the smooth flow-based approach, we use deep feature classification. We extract hypercolumn features [23] which we compute with VGG16 available in MatConvNet [43] to train a Gaussian-kernel support vector machine (SVM) classifier. We include a simple attention model that consists of applying this method in two stages. During the first stage, the result is used to localize the segmentation target. The segmentation is done using the localization result to focus the classification around the target.

Our evaluation first looks at the total time spent by workers on each task. We consider the crowdsourced segmentations every P frames as keyframes, from which we propagate the segmentation using each of the aforementioned methods. We then collect ten different scribbling results over each propagated frame using our scribble task. Given timings, we evaluate cost-effectiveness with varying scribble replications $R = 1, \ldots, 10$. One task assignment consisted of annotating the three candidate propagation intervals (P frames, each for all three propagation candidates). We used uniform keyframe samplings $P = 25, 10, 5, 3$ for an unbiased evaluation. The task reward was $0.015 per single frame scribbling.

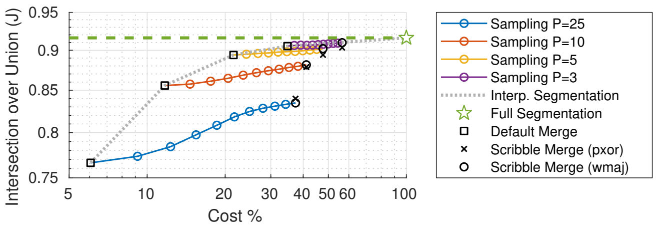


Figure 8. Evolution of the propagated quality with respect to the acquisition cost for different sampling intervals. The successive markers correspond to merging propagations using pixelwise majority (square), increasing scribbles replications using *wmaj* (circles) and full replication using *pxor* (cross). The costs are represented as the percentages to the naive full segmentation cost (star).

To evaluate the impact of the number of propagation methods, we additionally used three recent propagation techniques: Bilateral Video Segmentation [33] that propagates the segmentation using the bilateral grid; Video Propagation Networks [26] that uses a neural network with bilateral convolution layers; and the unsupervised technique FusionSeg [24] that uses objectness [25] to propose an object segmentation. Each of the corresponding propagations (for $P = 25, 10, 5, 3$) was acquired similarly to the original experiment, but only over the validation sequences of DAVIS (20 out of 50). Although FusionSeg [24] does not require a frame to propagate from, we still created tasks with sequences of corresponding length P so as to evaluate the impact of task load. Finally, because of limited space, we only show evaluation figures for the J metric; see the supplementary for the corresponding F and T figures.

The supplementary material also includes details about the full timing analysis, justifications for the brush sizes, smaller regularization, as well as an analysis of the scribble improvements.

Propagation Results

We first consider the time cost of both segmentation and scribble tasks, and then verify the positive impact of scribbles on the propagated segmentation quality. We look at how such increase in quality compares to a denser sampling of frames for individual segmentation with respect to the cost at different levels of replication. We then evaluate scribble design components including the use of different brush sizes, the impact of brush regularization, different merging strategies, and the merging order relative to scribble acquisition. Finally, we consider the impact of increasing the number of propagation methods and the task load.

Time Analysis and Time-Cost Tradeoffs

Figure 7 exposes the total cumulative task times using task prices of Table 2 for an approximate $7.0 hourly rate. The average per-frame segmentation takes 142.6 seconds, whereas per-frame per-method scribbles took only 2.5 seconds. The first obvious result is that our full replication $R = 10$ for scribbles goes beyond the cost of increasing the replication to the next P value.

The performance improvement that comes from using the scribbles during merge is shown in Figure 8, where we performed the evaluation on all 50 sequences of DAVIS [38] to report the average error metrics at each sampling interval

Initial brush	Mean J		Mean F	
	pxor	*wmaj*	*pxor*	*wmaj*
Large brush	0.8083	0.8362	0.8406	0.8556
Small brush	0.7981	0.8270	0.8288	0.8471

Table 3. Impact of the initial brush size. *pxor* refers to the corrective merge, and *wmaj* to the penalty-based merge. The mean values are over all sequences using a sampling $P = 25$. It appears that using a larger initial brush size is beneficial; using a smaller one seems to discourage some workers.

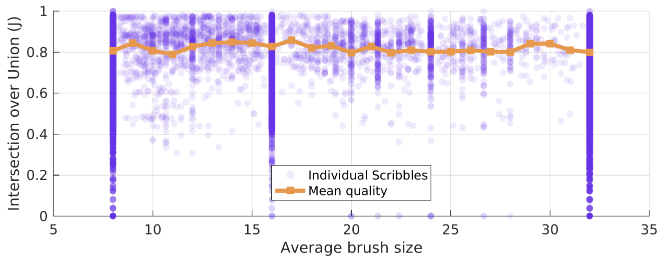


Figure 9. The joint distribution of brush sizes and corresponding result qualities (J only; see the supplementary material for F). Brush sizes included $b = 8, 16, 32$. Each scatter point corresponds to the average brush size of a single scribble, for all sequences with sampling $P = 25$.

$P = 25, 10, 5, 3$, with replications $R = 1, \ldots, 10$. The cost of each step comes from Table 2. Under our settings, cost efficiency was not achieved for most replications and samplings. We note that the penalty merging *wmaj* is more stable than the corrective merging *pxor*. The former always increases the quality whereas the later becomes detrimental as the sampling reaches small intervals ($P = 5$ and $P = 3$).

Impact of Brush Size

We evaluated the impact of the initial brush size and put this in perspective with the distribution of brush sizes that workers used given the quality of the result they contributed to. We provided three different brush sizes corresponding to radii of $b \in \{8, 16, 32\}$ screen pixels and re-ran the original scribble experiment at sampling intervals $P = 25$ with the initial brush size being the smallest this time in contrast to the largest being default. Selecting the smallest brush size initially led to a lower average brush size being used: $E[b] = 22.1$ when the initial brush size was $b_0 = 8$ versus $E[b] = 26.2$ when the initial brush size was $b_0 = 32$. However, it also led to a slightly lower quality as summarized in Table 3.

In Figure 9, we show the joint distribution of brush sizes and corresponding result quality. We can observe three peak concentrations of single brush sizes being used for $b = 8, 16$ or 32. Workers who use a mix of different sizes produce better results on average ($J = 0.82, F = 0.85$) than those using a single brush ($J = 0.80, J = 0.83$). However, they only accounted for 2.7% of the total scribble work (2,875 out of 105,641 valid assignments).

Impact of Brush Regularization

Brush regularization seems to have a consistent impact on the quality as shown in Figure 10. The best results are obtained with regularization $f \in [0.5, 0.8]$. Note also that the penalty-based merging strategy is generally more stable and the regularization has less impact on it. However, some of the best results are from the corrective strategy with a low regularization $f < 1$.

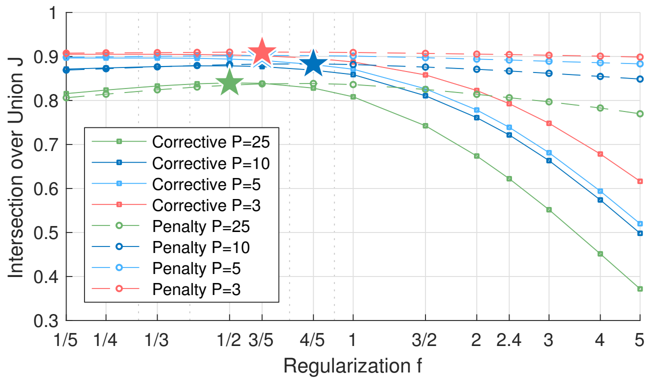


Figure 10. Impact of the brush regularization f on scribbles at different samplings P for both the corrective (*pxor*) and penalty-based merging (*wmaj*) strategies. For each sampling value, the best regularization and method are marked with a star.

Merging order	Regul'n f	Metric J	F	T
After scribbles	1	0.808	0.841	0.499
After scribbles	1/2	**0.840**	**0.869**	**0.489**
Before scribbles	1	0.749	0.818	0.817
Before scribbles	1/2	0.789	0.828	0.694
Mean (no scribble)		0.766	0.782	0.534

Table 4. Impact of the merging order using *pxor* at $P = 25$. Acquiring scribbles before merging leads to more information at merging time, which seems always beneficial. The best values are shown in boldface. The orange cells are the cases where using scribbles was detrimental.

Impact of Merging Order

Table 4 shows the effect of merging the multiple candidate propagations *before* and *after* the scribbles are requested. Merging afterward provides finer annotation capabilities as we request scribbles for multiple complementary candidates, which leads to better performance as expected. Interestingly, applying the scribbles without regularization is detrimental when applied after merging. This suggests that the scribbles are not sufficient as a fixing mechanism for a single segmentation. Instead, they can be used as a weighting mechanism when merging multiple segmentations.

Using Two Different Brushes

Table 5 compares the two-brushes scenario with the single-brush ones. Brush regularization does not really make sense since the workers directly interact with the segmentation in this scenario. In practice, it does not seem to have a big impact. Both J and F values are slightly lower than the single brush variants but not by a significant amount. On the contrary, the temporal stability is better.

Using Additional Propagation Candidates

Figure 11 shows the evolution of the quality with the increasing number of propagation methods used, for interval $P = 25$. For the scribble ensembles, we use the corrective merging strategy with regularization $f = 1/2$. As expected, using more methods increases the quality with a diminishing improvement. Note also that our scribble ensembles always outperform the default majority voting without scribbles.

Brush scenario	Regul'n f	Metric J	F	T
pxor	1/2	**0.840**	**0.869**	0.489
wmaj	1	0.836	0.856	0.468
2-brushes	1/2	0.833	0.850	0.467
2-brushes	1	0.835	0.846	**0.431**

Table 5. Comparisons of the two-brushes scenario with the two other single-brush ones. In terms of quality, the results are quite similar although slightly lower, except for the temporal stability T.

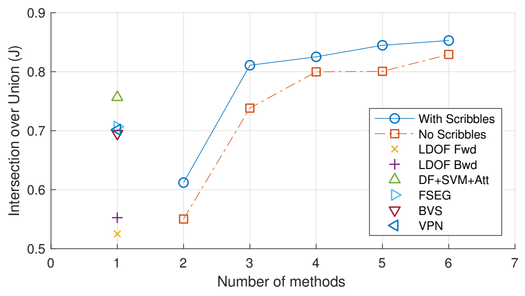


Figure 11. Evolution of the quality with the increasing number of propagation methods. The methods being additionally used are, in an increasing order: LDOF forward, LDOF backward, DF+SVM+Att, FSEG, BVS, and VPN. We handpicked these combinations of methods such that their complementedness is maximized. The evolution of the boundary accuracy (F) is similar and thus only provided in the supplementary material.

Impact of Task Load

The longer the scribble sequence, the more work needs to be done. Although the task reward was linearly proportional to the sequence length, humans have a limited budget of attention. Thus we evaluate the amount of work our workers did with respect to the task load (i.e. sequence length). For most of our scribble experiments, the sampling interval P is correlated with the propagation quality, which also leads to a different amount of work required for the intermediate frames. The main exception is the FusionSeg [24] method, and thus we use it to evaluate task load. Table 6 provides the quantitative results.

First, we analyze the total number of brush stroke vertices. While this metric contains the work of outlier scribbles, empty scribbles do not contribute and the amount of single point scribbles is not significant in comparison to the total number of vertices ($< 1\%$). Thus it is a reasonable proxy for the amount of work. The total number of vertices is somewhat similar to the average for task loads $P = 10, 5, 3$, whereas $P = 25$ generated about 23% fewer vertices. Beyond the amount of work, we considered the scribble coverage: (1) true positive scribble pixels and (2) false positive ones. True positive pixels are positive scribble pixels that cover propagated segmentation pixels that do not match the ground truth, whereas false positive pixels wrongly cover segmentation pixels that match the ground truth. The brush size was not regularized (i.e. $f = 1$) so as to match the worker's point of view. As expected from the number of vertices, the counts for $P = 25$ are also smaller. However, the ratio of true positive to false negative is the largest for $P = 25$. These results hint to the possibility that $P = 25$ was too much of a load for some workers, but they also show that workers who accepted those longer tasks tended to

Metric	Sampling P 25	10	5	3
Stroke vertices	1.0 M	1.5 M	1.3 M	1.3 M
Single vertices	1.6 k	3.6 k	4.3 k	3.9 k
Empty scribbles	9.4 k	9.5 k	6.7 k	8.6 k
True positive	87.9 M	104.1 M	130.6 M	136.0 M
False positive	193.3 M	280.2 M	386.4 M	316.1 M
TP / FP	0.46	0.37	0.34	0.43

Table 6. Quantitative evaluation of the task load for scribbles acquisition. Sampling $P = 25$ produced 23% fewer stroke vertices than the average, but it also achieved the highest precision. Sampling $P = 10$ produced 17% more stroke vertices, but it resulted in the worst precision.

do a more accurate work. Thus the task load may possibly be used as a quality filter.

DISCUSSIONS

We further discuss the key findings of our experiments here.

Best Ensemble Method Settings

Segmentation Ensembles: Negative annotation is an important tool for high-quality segmentation. High-quality workers seem to use it extensively. Clustering is not necessary, but it helps maintain quality when replication counts vary. Higher replication counts help, but three workers were often sufficient to create high-quality segmentations. Crowd-based reviews result in quality improvements, but an oracle reviewer enables higher-quality segmentation.

Scribble Ensembles: The penalty-based merging strategy *wmaj* generates more stable results with little dependency on the brush regularization or the initial segmentation quality. Our scribble annotations are not to be considered as fixing mechanisms, but as localized weighting mechanisms for merging multiple segmentation proposals. Using two brushes might be a good idea, but we do not have conclusive results about it yet. Using more diverse segmentation proposals always helps when possible. While using a high task load leads to lower amounts of work being done, it filters outliers. An optimal load was achieved with sequences of 25 frames in our experiments. This aligns with findings of Sigurdsson et al. [40] that larger loads may be preferable.

Limitations

Our main limitation is the cost overhead of the scribble task given the amount of work they involve. While our scribbles can increase the quality of the propagation ensembles and individually require much less time than segmentations, they still do so at a cost that is larger than increasing the segmentation sampling density, for a quality increase that is not as big. The main issue is that we apply them on every frame, resulting in a large factor ($P = 25$ leads to approximately $24 \times 3 = 72$ scribbles per keyframe). An interesting avenue for future work is to measure where scribbles are most needed so as to reduce the number of scribbles to segmentations.

Our review processes could be improved, notably about outliers. A common filter consists of Gold Standard tests. It requires defining good and bad results, which is time-consuming and requires a mean to detect bad segmentations or scribbles, which we do not have. Better merging strategies would involve estimating a confidence profile for the workers and their

Method	Metric J	F	T	W
LDOF	0.760	0.749	0.250	**0.204**
DF+SVM	0.764	0.782	**0.250**	0.192
DF+SVM+Att	**0.822**	**0.823**	0.317	0.198

Table 7. Comparisons between the scores W evaluated by crowd workers and the J, F, and T metrics, as in Table 1. All values are averages over all sequences of the DAVIS dataset. The score W is defined as the average number of positive evaluations for a frame segmentation.

skills [15]. Specifically, we could use superpixels to cluster brush strokes [12] and further evaluate worker confidence.

Do Workers Agree with Our Metrics?

We initially considered the task of deciding which of multiple propagation strategies is the best using the crowd. For a given propagation interval, we asked workers to select, for every propagation method, whether each frame propagation was good or bad after having shown them a selection of good and bad propagation results. The results of that experiment were not conclusive, but they are interesting as they seem to go against our validation metrics and thus we detail these here.

The propagation method using deep feature classification tends to produce good results in terms of pixel coverage, but the boundaries exhibit pixelation artifacts as illustrated in Figure 5. In this initial experiment, crowd workers often claimed (to our surprise) that the pixelation artifacts were worse than other results which are pixel-wise less accurate but have smoother boundaries (i.e. the optical flow based methods). This is detailed in Table 7, where we present evaluation scores W, measured as the average number of positive evaluations, over all DAVIS sequences propagated with our methods based on optical flow (a bidirectional variant selecting only the best out of both directions) and on deep feature classification (with and without the attention model). In practice, we always use the attention model, but we highlight that even without attention model, the propagation with deep features is more accurate according to the three metrics. Workers seemed to focus on the smoothness artifacts (which are stronger without attention model). While the relative preference (higher W) of LDOF over both other methods (DF+SVM$\pm$Att) is not necessarily significant, the results suggest that the human decision is not well correlated with the current metrics either. This calls for metrics that would match human perception better.

Segmentation Propagation for Video Matting

Our scribble-based propagation ensemble can easily be used for other application scenarios. We demonstrate it with an example of trimap propagation. In movie post-production, a common practice is the cut-out of an object from the background with proper opacity, subsequently composited with a novel background. Among typical techniques used for such composition are green-screen keying for constrained background (hence green-screen) and natural image matting for more demanding cases of unconstrained background (hence natural image). The typical input for matting is a trimap, which defines the opaque foreground and background regions, as well as the transition regions in-between, as seen in Figure 12. Generating trimaps for every frame in a video is a tedious and a time-consuming task [3, 46]. Our ensemble method can be

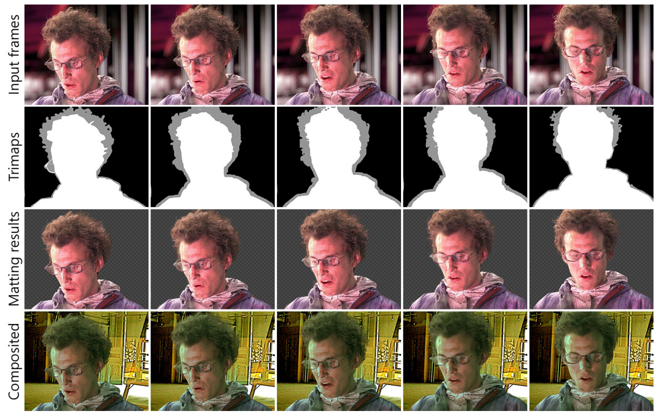


Figure 12. Given the trimaps for the first and last frames (*rightmost* and *leftmost* columns), the trimaps for the whole video can be generated using our method (second row). The trimaps are fed to a natural matting method [2] to get the foreground layer (third row), which is then used to generate a novel composition (fourth row). Note that the composited results are color graded. Images courtesy of (CC) Blender Foundation — mango.blender.org, Flickr user *chrstphre* and Aksoy et al. [4].

adapted to generate high-quality trimaps quickly for a video given the trimaps for the first and last frames.

We propagate the foreground, which is a conservative region that only includes opaque foreground regions, and the non-background part, which includes the foreground together with the opacity transition regions, separately using the proposed pipeline. We combine the two results in the end to get our final trimaps. In order to show an example of the full application scenario, we generated from these trimaps the foreground layer with semi-opacity properly considered, using information-flow matting [2], and composited them onto a novel background as presented in Figure 12.

CONCLUSION

We introduced two novel crowd-guided ensemble methods that combine multiple inputs from the crowd as well as automated algorithms to generate final segmentations that are better than any of their individual components. First, our results show that we can acquire image segmentations of higher quality by combining the work of multiple individuals. They confirm that negative polygon annotations are effective for crowdsourced segmentation. They also show that using our scribble-based ensemble method, we can delegate the review process of our oracle to the crowd to a certain extent. In practice, the oracle may well be required because rotoscoping work is often tied to some artistic decision that requires human validation. On the propagation side, our scribble-based ensembles are promising given their flexibility and the potential quality increase over naive ensembles, although they do not achieve cost efficiency compared to increase segmentation rates. Finally, the web interfaces, tools, and data of our experiments will be made public for future research.

Acknowledgements

Changil Kim was supported by a Swiss National Science Foundation fellowship P2EZP2 168785. We thank Zoya Bylinskii for her help with the paper.

REFERENCES

1. Aseem Agarwala, Aaron Hertzmann, David H. Salesin, and Steven M. Seitz. 2004. Keyframe-based Tracking for Rotoscoping and Animation. *ACM Trans. Graph.* 23, 3 (2004), 584–591.

2. Yağız Aksoy, Tunç Ozan Aydın, and Marc Pollefeys. 2017. Designing Effective Inter-Pixel Information Flow for Natural Image Matting. In *Proc. CVPR*.

3. Yağız Aksoy, Tunç Ozan Aydın, Marc Pollefeys, and Aljoša Smolić. 2016. Interactive High-Quality Green-Screen Keying via Color Unmixing. *ACM Trans. Graph.* 35, 5 (2016), 152:1–152:12.

4. Yağız Aksoy, Tunç Ozan Aydın, Aljoša Smolić, and Marc Pollefeys. 2017. Unmixing-Based Soft Color Segmentation for Image Manipulation. *ACM Trans. Graph.* 36, 2 (2017), 19:1–19:19.

5. Xue Bai, Jue Wang, David Simons, and Guillermo Sapiro. 2009. Video SnapCut: Robust Video Object Cutout Using Localized Classifiers. *ACM Trans. Graph.* 28, 3 (2009).

6. Sean Bell, Kavita Bala, and Noah Snavely. 2014. Intrinsic Images in the Wild. *ACM Trans. Graph.* 33, 4 (2014).

7. Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. 2013. OpenSurfaces: A Richly Annotated Catalog of Surface Appearance. *ACM Trans. Graph.* 32, 4 (2013).

8. Sean Bell, Paul Upchurch, Noah Snavely, and Kavita Bala. 2015. Material Recognition in the Wild with the Materials in Context Database. *Proc. CVPR* (2015).

9. Steve Branson, Kristjan Eldjarn Hjorleifsson, and Pietro Perona. 2014. Active annotation translation. In *Proc. CVPR*.

10. Benjamin Bratt. 2011. *Rotoscoping: Techniques and Tools for the Aspiring Artist*. Taylor & Francis.

11. Thomas Brox and Jitendra Malik. 2011. Large displacement optical flow: descriptor matching in variational motion estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* 33, 3 (2011), 500–513.

12. Ferran Cabezas, Axel Carlier, Vincent Charvillat, Amaia Salvador, and Xavier Giro-i Nieto. 2015. Quality control in crowdsourced object segmentation. In *Image Processing (ICIP), 2015 IEEE International Conference on*.

13. Sergi Caelles, Kevis-Kokitsi Maninis, Jordi Pont-Tuset, Laura Leal-Taixé, Daniel Cremers, and Luc Van Gool. 2016. One-Shot Video Object Segmentation. *CoRR* abs/1611.05198 (2016).

14. Peng Dai, Mausam, and Daniel S. Weld. 2010. Decision-theoretic control of crowd-sourced workflows. In *Proc. AAAI*.

15. Alexander Philip Dawid and Allan M Skene. 1979. Maximum likelihood estimation of observer error-rates using the EM algorithm. *Applied statistics* (1979).

16. Jia Deng, Olga Russakovsky, Jonathan Krause, Michael S Bernstein, Alex Berg, and Li Fei-Fei. 2014. Scalable multi-label annotation. In *Proc. SIGCHI*.

17. Thomas G Dietterich and others. 2000. Ensemble methods in machine learning. *Multiple Classifier Systems* 1857 (2000).

18. Suyog Dutt Jain and Kristen Grauman. 2013. Predicting sufficient annotation strength for interactive foreground segmentation. In *Proc. ICCV*.

19. Anhong Guo, Xiang'Anthony' Chen, Haoran Qi, Samuel White, Suman Ghosh, Chieko Asakawa, and Jeffrey P Bigham. 2016. VizLens: A robust and interactive screen reader for interfaces in the real world. In *Proc. UIST*.

20. Danna Gurari and Kristen Grauman. 2017. CrowdVerge: Predicting If People Will Agree on the Answer to a Visual Question. In *Proc. SIGCHI*.

21. Danna Gurari, Mehrnoosh Sameki, and Margrit Betke. 2016. Investigating the Influence of Data Familiarity to Improve the Design of a Crowdsourcing Image Annotation System. In *Proc. HCOMP*.

22. Kotaro Hara, Jin Sun, Robert Moore, David Jacobs, and Jon Froehlich. 2014. Tohme: detecting curb ramps in google street view using crowdsourcing, computer vision, and machine learning. In *Proc. UIST*.

23. Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. 2015. Hypercolumns for object segmentation and fine-grained localization. In *Proc. CVPR*.

24. Suyog Jain, Bo Xiong, and Kristen Grauman. 2017a. FusionSeg: Learning to combine motion and appearance for fully automatic segmentation of generic objects in videos. *Proc. CVPR* (2017).

25. Suyog Jain, Bo Xiong, and Kristen Grauman. 2017b. Pixel Objectness. *arXiv:1701.05349* (2017).

26. Varun Jampani, Raghudeep Gadde, and Peter V. Gehler. 2017. Video Propagation Networks. In *Proc. CVPR*.

27. Aniket Kittur, Jeffrey V. Nickerson, Michael Bernstein, Elizabeth Gerber, Aaron Shaw, John Zimmerman, Matt Lease, and John Horton. 2013. The future of crowd work. In *Proc. CSCW*.

28. Shrenik Lad and Devi Parikh. 2014. Interactively guiding semi-supervised clustering via attribute-based explanations. In *Proc. ECCV*.

29. Gierad Laput, Walter S Lasecki, Jason Wiese, Robert Xiao, Jeffrey P Bigham, and Chris Harrison. 2015. Zensors: Adaptive, rapidly deployable, human-intelligent sensor feeds. In *Proc. SIGCHI*.

30. Wenbin Li, Fabio Viola, Jonathan Starck, Gabriel J. Brostow, and Neill D.F. Campbell. 2016. Roto++: Accelerating Professional Rotoscoping using Shape Manifolds. *ACM Trans. Graph.* 35, 4 (2016).

31. Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft COCO: Common Objects in Context. In *Proc. ECCV*.

32. Yao Lu, Xue Bai, Linda Shapiro, and Jue Wang. 2016. Coherent parametric contours for interactive video object segmentation. In *Proc. CVPR*. 642–650.

33. Nicolas Märki, Federico Perazzi, Oliver Wang, and Alexander Sorkine-Hornung. 2016. Bilateral Space Video Segmentation. In *Proc. CVPR*.

34. Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. 2014. The Role of Context for Object Detection and Semantic Segmentation in the Wild. In *Proc. CVPR*.

35. Genevieve Patterson and James Hays. 2016. COCO Attributes: Attributes for People, Animals, and Objects. In *Proc. ECCV*.

36. Genevieve Patterson, Grant Van Horn, Serge J Belongie, Pietro Perona, and James Hays. 2015. Tropel: Crowdsourcing Detectors with Minimal Training. In *Proc. HCOMP*.

37. Federico Perazzi, Anna Khoreva, Rodrigo Benenson, Bernt Schiele, and Alexander Sorkine-Hornung. 2016a. Learning Video Object Segmentation from Static Images. In *Proc. CVPR*.

38. Federico Perazzi, Jordi Pont-Tuset, Brian McWilliams, Luc Van Gool, Markus Gross, and Alexander Sorkine-Hornung. 2016b. A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation. In *Proc. CVPR*.

39. Brian L Price, Bryan S Morse, and Scott Cohen. 2009. Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues. In *Proc. ICCV*.

40. Gunnar A. Sigurdsson, Olga Russakovsky, Ali Farhadi, Ivan Laptev, and Abhinav Gupta. 2016. Much Ado About Time: Exhaustive Annotation of Temporal Data. In *HCOMP*.

41. Chong Sun and Huchuan Lu. 2017. Interactive video segmentation via local appearance model. *IEEE Trans. Circuits Syst. Video Technol.* 27, 7 (2017), 1491–1501.

42. Yi-Hsuan Tsai, Ming-Hsuan Yang, and Michael J. Black. 2016. Video Segmentation via Object Flow. In *Proc. CVPR*.

43. Andrea Vedaldi and Karel Lenc. 2015. MatConvNet – Convolutional Neural Networks for MATLAB. In *Proc. ACM MM*.

44. Sudheendra Vijayanarasimhan and Kristen Grauman. 2014. Large-scale live active learning: Training object detectors with crawled data and crowds. *Int. J. Comput. Vision* 108, 1-2 (2014), 97–114.

45. Carl Vondrick, Donald Patterson, and Deva Ramanan. 2012. Efficiently Scaling up Crowdsourced Video Annotation. *Int. J. Comput. Vision* (2012).

46. Jue Wang and Michael F. Cohen. 2005. An iterative optimization approach for unified image segmentation and matting. In *Prov. ICCV*.

47. Jenny Yuen, Bryan Russell, Ce Liu, and Antonio Torralba. 2009. Labelme video: Building a video database with human annotations. In *Proc. ICCV*.

48. Peng Zhang, Jiuling Wang, Ali Farhadi, Martial Hebert, and Devi Parikh. 2014. Predicting failures of vision systems. In *Proc. CVPR*.