Gamifying Video Object Segmentation

Simone Palazzo, Concetto Spampinato, *Member, IEEE* and Daniela Giordano, *Member, IEEE*

✦

Abstract—Video object segmentation can be considered as one of the most challenging computer vision problems. Indeed, so far, no existing solution is able to effectively deal with the peculiarities of real-world videos, especially in cases of articulated motion and object occlusions; limitations that appear more evident when we compare their performance with the human one. However, manually segmenting objects in videos is largely impractical as it requires a lot of human time and concentration. To address this problem, in this paper we propose an interactive video object segmentation method, which exploits, on one hand, the capability of humans to identify correctly objects in visual scenes, and on the other hand, the collective human brainpower to solve challenging tasks. In particular, our method relies on a web game to collect human inputs on object locations, followed by an accurate segmentation phase achieved by optimizing an energy function encoding spatial and temporal constraints between object regions as well as human-provided input. Performance analysis carried out on challenging video datasets with some users playing the game demonstrated that our method shows a better trade-off between annotation times and segmentation accuracy than interactive video annotation and automated video object segmentation approaches.

Index Terms—Interactive video annotation, Games with a purpose, Human in the Loop, Spatio-temporal superpixel segmentation

1 INTRODUCTION

THE generation and collection of massive amount of videos has become an easy task due to the progress in low-cost digital imaging systems as well as storage services. Indeed, every day several petabytes of videos are routinely generated for disparate applications ranging from video-surveillance to news broadcasting to entertainment. This is also highlighted in the recent "Forecast and Methodology 2014-2019" report[1] by CISCO that has estimated that consumer internet video traffic will be 80 percent of all consumer Internet traffic in 2019. Nevertheless, this video data deluge needs automated methods able to extract meaningful information for data indexing, preservation and understanding, since it is unrealistic and unfeasible (as stated in the same CISCO report, *it would take an individual over 5 million years to watch the amount of video that will cross global IP networks each month in 2019*) to assume people can do this work manually. This is the main reason why all the existing video datasets have only a few frames annotated. One of the upstream modules for video understanding is object segmentation, which aims at discriminating accurately foreground objects from the background. There is a large (past and present) bulk of literature of methods for video object segmentation. Background modeling/subtraction [1], [2], motion analysis [3], [4], object ranking [5], [6] and clustering point tracks [7], [8] and, recently, combination of CNN-based moving object detectors [9] are among the most common methods. However, the accuracy and performance of these techniques are still not satisfactory, especially in cases of articulated motion, cluttered scenes and object occlusions. Therefore, so far, there exist no valid alternatives to the classic automated video segmentation methods and the only possible solution might be to include effectively and efficiently humans in the analysis and learning process. "Human in the loop" is a recent trend in machine learning which tries to learn discriminative patterns by proactively involving people in the annotation process. This is the case of "games with a purpose" that channel collective human brainpower through computer games [10]. The underlying idea is to engage people in solving unconsciously complex tasks while playing computer games. The combination of these games to crowdsourcing strategies may be an extremely powerful tool to solve tasks at large scale. While there exist several games with purpose to support automated image analysis [11], [12], [13], [14], [15] and some for video tagging [16], [17] (which, however, share the same philosophy of the image annotation ones), to the best of our knowledge, none have been adopted for video object segmentation, which would particularly benefit from this approach as annotating videos, in terms of object segmentation, requires much more human time and concentration than identifying image classes. Under this scenario, in this paper we propose a human-guided video object segmentation method, built upon a web game and able to effectively and accurately extract moving objects from videostreams by greatly reducing human intervention. The contribution of the paper is threefold:

- First, we present and release a web game to collect human input (in the form of clicks) that can be used with any kind of videos, thus representing a powerful tool for computer vision scientists (and not only) to get their own videos automatically annotated;
- We propose an interactive video object segmentation approach based on the optimization of an

- *S. Palazzo, C. Spampinato and D. Giordano are with the Department of Electrical, Electronics and Computer Engineering, University of Catania, Italy.*
  E-mail: see http://perceive.dieei.unict.it

1. http://www.cisco.com/c/en/us/solutions/collateral/service-provider/ip-ngn-ip-next-generation-network/white_paper_c11-481360.html

energy function which is able to encode spatio-temporal constraints between object regions as well as human-provided priors. The method can be combined with other sources of human input (e.g., eye-gaze data [18]) to annotate automatically videos.

- We demonstrate that the collective action of players, despite providing noisy and inaccurate data, results in better segmentation accuracy – with much less human effort – than state-of-the-art automated video object solutions as well as interactive video annotation methods.

2 RELATED WORK

Our work shares the same end goal of automated video object segmentation [1], [2], [19], [20], [21], [22], [23], [24], but our approach is more inline with research that places humans in the loop (including games with purpose) [11], [12], [13], [14], [15], [16], [17], [25], [26], [27], [28], [29], [30], [31], [32] and interactive video segmentation [7], [33], [34], [35], [36], [37], [38], [39].

Unsupervised video segmentation has gained a lot of attention in the last decades [2], [19], [20], [21], [22] and recently it has been thought mainly in terms of spatio-temporal superpixel modeling [1], [23], [24]. The key idea behind these methods is the one of grouping pixels which are appearance- and motion-wise–consistent. Despite the performance increase due to superpixel segmentation, all these methods suffer from oversegmentation, especially in cases of camera motion and object occlusions. Therefore, manual or semi-manual video annotation can be considered the only reliable way for obtaining precise object segmentation, but as argued earlier, this tedious and labor-intensive process is extremely costly and unfeasible at large scale. Semi-supervised video segmentation approaches [7], [33], [34], [35], [36], [37], that differently from the semi-supervised image segmentation ones (e.g., the popular Grabcut [40]) have not received much attention, usually require a short intervention by humans in terms of object annotations that are then propagated automatically over time. Most of these methods rely either on optical flow [34] or on temporal connections [35], [37] between frames usually modeled by Markov chains, but they work well only in simple cases failing mainly in precisely representing object boundaries. In [39], the authors propose an interactive video object segmentation method using a 3D graph-cut–based segmentation followed by a tracking-based local refinement. In [36] video object segmentation is formulated as a spatio-temporal MRF optimization problem, with a cost function including user input, motion and appearance cues, spatio-temporal consistency similarly to the one proposed in this work. In addition, superpixel segmentation is also largely employed as it allows us to reduce processing time ensuring at the same time spatio-temporal coherency among pixels. In some work [41], temporal linking between superpixels is done manually. Despite these methods are able to alleviate human effort for video annotation, at large scale they are ineffective and still time-consuming for human operators. Another option to support low-level computer vision tasks is to understand how human perform them and to seek how human inference/reasoning can be integrated into computer programs. Examples are the ones that ask people to provide explicitly annotation rationales [42] or to elicit the visual features employed to discriminate between image/object classes [25], [43]. Nevertheless, unlike computers, humans need incentives, either monetary or for entertainment, to carry out specific tasks. Under this scenario, on-line games represent an effective mechanism to involve people in solving challenging problems. Two of the most common approaches exploiting web-games for collecting human feedback for machine learning methods are the ESP Game [13] and Peekaboom [15]. Both approaches use the collective intelligence of human brains for gathering key information for image classification. Since their release, many thousands of people have played them, generating millions of labels. However, these games are devised only for image analysis and, moreover, cannot be played by everyone: for instance, children, who usually are very passionate with games, would have difficulty in getting engaged by them. In addition, to the best of our knowledge, there exist no games employed for supporting video object segmentation, except the one used in this work.

The approach proposed in this paper draws inspiration from both interactive video object segmentation approaches and human-computation using web-games combining both strategies in a smart way for accurate object segmentation in videos at large scale. More specifically, we propose an interactive video segmentation approach formulated as a spatio-temporal superpixel labeling by taking into account user input and spatio-temporal consistency of motion and appearance features. In addition, user feedback is gathered through a web game in the form of clicks, instead of strokes (as in most of the interactive video object segmentation methods), leveraging on multiple users to obtain more consistent and structured feedback for automated segmentation. Also, the web-game is designed to being playable by any person of any age, thus increasing its possible audience (and with it the amount of gathered data) and improving the accuracy of the generated object segmentations.

3 METHOD

The proposed interactive video object segmentation approach can be seen as a two-step spatio-temporal MRF optimization problem: the first one with a cost function exploiting spatial information at the frame level and encoding user input and appearance cues in order to extract homogeneous object regions in video frames; and the second one enforcing spatio-temporal consistency between the segmented object regions in consecutive frames, thus refining the preliminary segmentation. Three are the main modules of the whole approach:

- *The game*: The starting point of the whole process, our game is thought to gather user clicks in correspondence of objects of interest in videos. The game is designed to be challenging and competitive, so that users are encouraged to play: while this helps keeping the competition between users, game difficulty often reflects on the noisiness of the generated data.
- *Superclick extraction*: The initial stage of our algorithm converts the noisy set of clicks into a set of more accurate "clicked superpixels", or *superclicks*. Posing the

problem in terms of superpixels rather than pixels 1) reduces the numerical complexity of the task and, 2) enforces spatial coherency between clicked object regions. On top of this viewpoint, we identify and group together superclicks through MRF optimization.

- *Temporal smoothing*: Single-frame superclick extraction produces a fairly accurate segmentation of the objects in the scene, however it ignores temporal consistency between frames which can be exploited to further improve the segmentation. Based on the superclicks extracted from a span of consecutive frames, a three-dimensional (across time) MRF is designed in order to transfer information on the labels assigned to corresponding superpixels at different frames.

3.1 The Game

We reused the game presented in [44] (where users had to perform a similar task but for a different objective, i.e., ground-truth generation) by adapting and modifying it according to our objectives: players are instructed to click on moving objects in a set of videos (one for each game level). Each correct click awards points, and each level is successfully completed if the user sums up a certain amount of points (increasing by level).

User interface. Fig. 1 shows an example of a typical in-game screenshot. The video for the current level is, of course, the most important element of the interface and takes up most of the space; the current score obtained by the user is shown at the top; the remaining time before the level's end is shown on the top-left corner (indicated by the OXYGEN icon—a legacy from the original underwater-oriented application of the game), and the number of points needed to pass the current level is at the bottom of the screen. The mouse cursor is shaped like a camera reticle, and at each click the taken "photo" is shown at the bottom-left corner of the screen (this is done for future object classification purposes, which are beyond the scope of this paper). When the user clicks correctly on a target, points are awarded, shown as upward-floating bubbles ("+81" in the example image). Finally, further option buttons are shown at the top-right corner of the screen.

Levels. Each level in the game is associated to an input video, which is supposed to be at least 30 seconds long at 10 frames per second (if longer, only the first 30 seconds will be shown). In order to pass a level, a certain amount of points must be scored, starting from 4000 at the first level, and increasing by 2000 at each successive level. As the game is actually relatively simple, this increase represents the main challenge, since it makes it more and more difficult to achieve the required points.

Level-video association is done randomly, i.e., in different game sessions, the video ordering is never the same in order to avoid players to know in advance where objects might be located. This is necessary since in games, players often tend to maximize their scores also using tricks.

One related issue was the *saliency bias* of some objects with respect to others, which caused users to click always on the same objects (the most salient ones) in a scene even if

Fig. 1. In-game screenshot of the user interface.

several others were present. To reduce this phenomenon, we applied an *inhibition of return* mechanism by blurring videos in areas where clicks (by all users) accumulate: this reduced the saliency of underlying objects and led users to click on other objects in the scene. Video blurring was performed using all gathered clicks, and not just the current user's, although saliency is partly a subjective process: we found this also helped to avoid gathering too many clicks on objects for which we already had enough data (see paragraph 4.4.2). Fig. 2 shows an example of the click distribution in a frame and the corresponding blurred version.

Points. As in any gamification process, it is necessary to pose the task as a competitive one, providing the users with a feedback on how good they are with respect to their previous results or their friends. We employ a point-based system to reflect users' performance on the game, and keep an all-time ranking of the best scores. Points are awarded by clicking correctly on an object of interest, depending on the size of the object and on previous clicks: bigger objects are awarded more points, but successive clicks in the same area earn the user less and less points, according to the formula:

$$P^+ = \frac{A}{20}\left(1 - \frac{t}{10}\right) \tag{1}$$

where P^+ is number of points earned for a correct click, A is the area in pixels of the clicked object, and t is the number of consecutive clicks within a $30{\times}30$ pixels region. In practice, $\frac{A}{20}$ is the maximum score awarded for a click on a certain object, but this score is progressively reduced if the user keeps clicking on the same spot: when salient objects are in the scene, this reduction forces users to vary their clicking pattern to get more points, while at the same time helps to provide data on as many objects as possible. Conversely, users are subtracted points if they click too far from the objects in the visualized frame; the penalty is computed as:

$$P^- = 20t \tag{2}$$

where P^- is the amount of points subtracted to the current score due to a wrong click, and t is the number of consecutive clicks falling further than 200 pixels from the

Fig. 2. **Left**: user clicks (blue dots) in a frame; **Right**: saliency inhibition by blurring clicked regions.

closest correct object. Penalties prevent users from clicking randomly across the frame and force them to be as more accurate as possible.

However, to award points to players we need object segmentation (not necessarily highly accurate) on the input videos to tell whether clicks hit or miss objects. In order to have a reference signal — *score video segmentation*— according to which we assign points to players, we use the *output of the system itself*. When the system is first set up and no data is available yet, the initial video object segmentation is obtained by running a classic background modeling method ([19] in our case); although in the beginning this may not be enough to cover all and only objects in the scene, it still provides an adequate base for setting the game up. After users have started to play, the object segmentation is simply updated based on users' clicks by running the algorithm presented in this paper. It is not strictly necessary for *score video segmentation* to be extremely accurate: scores are only provided for the benefit of users, in order to keep them interested by means of competition.

Click quality. We also estimate the "quality" (in the sense of "accuracy of clicks with respect to objects") of the data provided by users while playing the game. Quality scores are computed on a per-level and per-user basis as the fraction of user clicks hitting the objects in the level. We assume that all clicked pixels in a game level by a user gets the same quality score computed as above. We could have computed a global quality score for a single game (i.e., the sequence of levels a user plays before completing the game) or for the user, however different levels may return completely different quality scores even within the same game session, due to each video's scene and object characteristics, so a global score would become too generic to describe individual click quality in a level's context.

3.2 Superclick Extraction

The clicks collected through the web game are used to extract information on the location of objects in the scene for each video frame and to carry out a preliminary object segmentation. We pose the problem as a binary segmentation task (background and foreground) by means of the minimization of an energy function defining the cost of a segmentation. Like some of the most recent methods for video object segmentation [1], [23], [24], we use *superpixels* (computed by SLIC [45]) as basic image parts instead of pixels as they provide two main advantages: 1) reducing the number of variables greatly speeds up the minimization algorithm (the number of variables is scaled down by a factor of 30-50, depending on superpixel settings); 2) the initial segmentation provided by superpixels is usually effective in detecting edges, which allows to simply focus on finding the optimal aggregation, taking boundary detection for granted.

The first processing step, given our target frame F, consists of *superclick extraction*, where a *superclick* is the intuitive extension of the concept of clicks to superpixels. This step is necessary to be able to pose the problem in terms of superpixels only, by "converting" point data (e.g., clicks) to superpixel-oriented ones. Of course, the principle behind this operation is that superpixels containing clicked pixels should be more likely to be marked as superclicks, which are then converted into constraints for MRF optimization. However, clicks are generally noisy, thus other factors, such as click density, click quality (as defined previously), closeness to other clicked superpixels need to be taken into account for superclick identification.

Before explaining how superclicks are computed, a more basic question is: *what clicks should we use to analyze a certain frame?* Depending on video frame rate and target speed, users' reaction times may introduce a delay which results in a shift between the frame at which the user clicks on an object and the frame at which the user *intended* to click. Fig. 3 shows a few examples: it is possible to notice that the delay effect is more visible on some videos than others according to mainly objects' speed. Since this issue involves complex biological phenomena [46], which are out of the scope of the paper, we adopt a simple but effective empirical approach: we assume that all clicks are delayed by a constant number of frames for all videos. In detail, the results in Sect. 4 shows that shifting all clicks back by 2 frames (although the optimal delay may vary from 1 to 4 frames which depends on several factors, one above all the video frame rate) represents a good trade-off between accuracy and complexity.

Let $C = \{c_1, c_2, \ldots, c_{n_C}\} = \{(x_1, y_1), (x_2, y_2), \ldots, (x_{n_C}, y_{n_C})\}$ be the players' clicks for frame F, with corresponding quality scores $Q = \left\{ q_{c_1}, q_{c_2}, \ldots, q_{c_{N_C}} \right\}$ (each click gets the quality score assigned to the user who did it on a per-level basis). We define a graph-representable energy function [47] over

Fig. 3. Due to users' reaction times, clicks may be delayed with respect to the "intended" frame. It is possible to notice that this phenomenon may be more or less evident even within the same image, depending not only on the user but also on the objects in the scene.

the set of F's superpixels $S = \{s_1, s_2, \ldots, s_{n_S}\}$, with a cost function able to model the "clickedness" of each superpixel independently, and at the same time, to enforce constraints on visual smoothness and click continuity. Our main assumptions are:

1) Superpixels containing a large number of clicks should be marked as superclicks (and vice versa), i.e., they can be seen as hard constraints for segmentation.
2) Clicked pixels should be weighted by the relative quality when evaluating their contribution to a superclick.
3) Unclicked superpixels which are close to clicked and visually-similar superclicks should be marked as superclicks as well, since they are likely to belong to the same object.
4) Isolated clicked superpixels (even if in small groups) should be ignored as being likely noise.

Translating these assumptions into energy potentials, we obtain the following cost function for energy optimization:

$$E_1(\mathcal{L}) = \alpha_1 \sum_{s \in S} V_1(s, l_s, C) + \sum_{(s_1, s_2) \in \mathcal{N}(S)} V_2(s_1, s_2, l_{s_1}, l_{s_2})$$

(3)

where $\mathcal{L} = \left\{ l_{s_1}, l_{s_2}, \ldots, l_{s_{n_S}} \right\}$ is the superclick label assignment (l_{s_i} is the binary superclick label for superpixel s_i), $\mathcal{N}(S)$ is the set of pairs of neighbor superpixels (that is, having part of boundary in common; we will also use the notation $\mathcal{N}(s)$ to denote the set of neighbors of the single superpixel s), and α_1 is a weighing factor.

Unary potential V_1 models whether superpixel s is likely to be a superclick or not. This "likeliness" depends on the number and quality of clicks inside the superpixel's region

and on the vicinity to clicked superpixels[2]. Therefore, V_1 is given by two contributions:

- *Clickedness* K_s: the more (high-quality) clicks a superpixel has received, the more it is likely to be a good candidate superclick. The clickedness score K_s for superpixel s is:

$$K_s = \underbrace{\frac{|C \cap s|}{\max_{t \in S} |C \cap t|}}_{\text{(4a)}} \underbrace{\frac{1}{|C \cap s|} \sum_{c \in C \cap s} q_c}_{\text{(4b)}} = \frac{\sum_{c \in C \cap s} q_c}{\max_{t \in S} |C \cap t|}$$

(4)

where $C \cap s$ is the set of clicks hitting superpixel s and $|\cdot|$ is set cardinality. The first (unreduced) version explains more clearly what this formula is meant for: Eq. term (4a) indicates how many clicks, superpixel s contains with respect to the superpixel containing most clicks in the processed frame; Eq. term (4b) is, instead, the average quality of clicks inside s, and encodes quality information in the score. The way this score is computed thus addresses items 1 and 2 of the above design principles.

- **Proximity to clicked superpixels** V_s: if s has not received many clicks but is close to superpixels which did, we might want to take it into consideration as a potential superclick. Of course, being close to clicked superpixels by itself is not enough: any superpixel just outside an object's boundary satisfies this requirement; this issue will be addressed by the pairwise potential V_2.

Our proximity score V_s is computed as the fraction of neighbor superpixels with clickedness score $K_{s_n} > 0.5$, with $s_n \in \mathcal{N}(s)$:

$$V_s = \frac{|\{s_n \in \mathcal{N}(s) : K_{s_n} > 0.5\}|}{|\mathcal{N}(s)|}$$

(5)

Analogously, if s gets enough clicks but is isolated, V_s will be low and K_s won't suffice to label it as a superclick. Thus, V_s balances items 3 and 4 of our design principles.

- **Unclicked regularizer**: the point of introducing the V_s score is to allow a superpixel with few or no clicks to be labeled as superclick if its neighborhood hints that it should; however, if an unclicked superpixel is not adjacent to any clicked superpixels, its V_1 potential is zero, which is something we want to avoid. Consider, for example, the case of an object consisting of a large uniform region with a non-uniform users' click distribution (which is actually often the case, as users tend to click at the center of objects): by setting unclicked superpixels to a low (but not null) potential, we allow labels to "spread" from superclicks (as per item 3 of our design principles above)—as long as uniformity

---

2. The reader might think that "vicinity to clicked superpixels" should be modeled as a pairwise potential, rather than unary. In fact, it should be modeled as unary because it is not an indication of whether two elements should be assigned the same label (which is what pairwise potentials represent); instead, it uses local information to indicate whether that item, *individually*, is more likely to be assigned to a specific label (1 for "superclick" or 0 for "not superclick")

requirements, defined by potential V_2, apply.

For this reason, we add a constant U_s term to the V_1 potential, which should be small enough not to "push" too much toward the "superclick" label (since clickedness and vicinity clues suggest it should not be), but not so small that it cannot ever be labeled as such.

The definitions of K_s, V_s and U_s have been chosen so that the sum of those terms (clipped to 1 if necessary) can be interpreted as the probability that superpixel s belongs to class "superclick", $P_{s,1} = P(l_s = 1|C,S) = \min(K_s + V_s + U_s, 1)$. Similarly, the complementary probability $P_{s,0} = P(l_s = 0|C,S) = 1 - P_{s,1}$ is the probability that s is "not a superclick". In the energy function, V_1 is meant to represent the cost of assigning a certain label to each superpixel: such costs can be computed as the negative log-likelihood of the two probabilities above:

$$V_1(s, l_s, C) = \begin{cases} -\log P_{s,1} & \text{if } l_s = 1 \\ -\log P_{s,0} & \text{if } l_s = 0 \end{cases} \quad (6)$$

Pairwise potential V_2 is the cost of assigning different labels to two adjacent superpixels s_1 and s_2: ideally, it should be large for "similar" superpixels (so that they are assigned the same label) and small for superpixels for which no evidence exists that they should belong to the same class. Although in general this function could depend on the specific labels being assigned (so that, for example, the cost of assigning labels ($l_{s_1} = 1, l_{s_2} = 0$) might be different than the cost of assigning labels ($l_{s_1} = 0, l_{s_2} = 1$)), in our case we focus only on estimating the optimal separation point between the "superclick"/"non-superclick" regions, based on visual similarity.

Therefore, potential V_2 is simply expressed as follows:

$$V_2(s_1, s_2, l_{s_1}, l_{s_2}) = \exp\left[-\beta_1 \chi^2(H_{s_1}, H_{s_2})\right] \mathcal{I}(l_{s_1} \neq l_{s_2}) \quad (7)$$

where $\chi^2(\cdot, \cdot)$ is the Chi-square distance, H_{s_i} is the RGB color histogram of superpixel s_i, β_1 is a constant, and $\mathcal{I}$ is an indicator function which returns 1 if the arguments is true, and 0 otherwise (this ensures that V_2 is a submodular function, thus making the whole energy function graph-representable [47]). Using a simple similarity measure such as the color histogram has a twofold justification: 1) by construction, superpixels have very little internal structure, so using more complex descriptors is unnecessary; 2) since the function has to be evaluated for all pairs of adjacent superpixels, it is important to perform as efficient operations as possible, in order to keep computation times reasonable.

Once $E_1(\mathcal{L})$ has been minimized by means of graph cut, the extracted superclicks already provide a good approximated segmentation of the objects of interest in the scene, as shown by the examples in Fig. 4. Nevertheless, output images at this stage can show segmentation errors, e.g., holes, oversegmentations, etc, and further processing by taking into account motion information is carried out to refine the obtained segmentation masks.

3.3 Temporal Smoothing

The superclick extraction step turns a set of noisy clicks into a set of spatial coherent superclicks per frame, but ignores
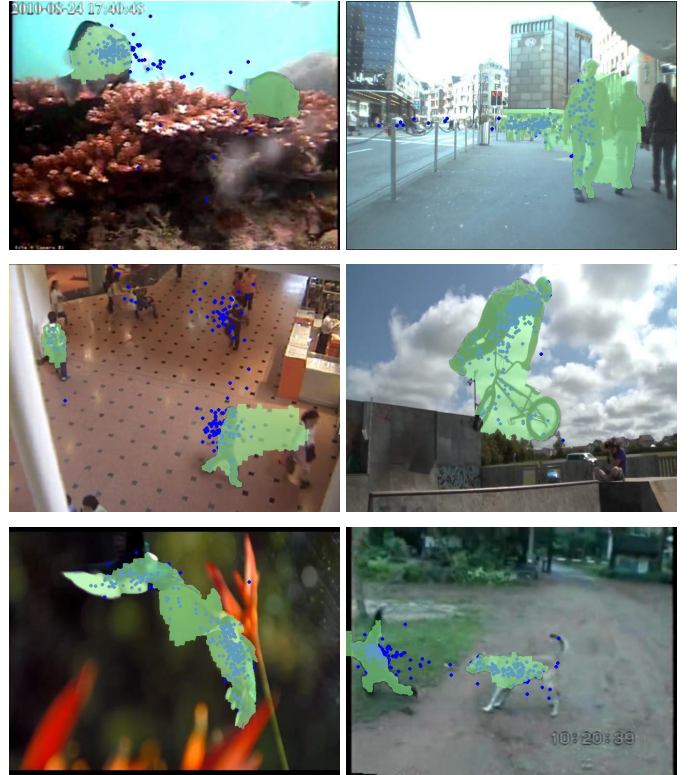

Fig. 4. Output examples for superclick identification: blue dots are users' clicks while green regions show the yielded segmentation masks. Segmentation refinement is carried out by including temporal constraints.

any temporal information which, instead, is necessary in video streams. Therefore, the next step for segmentation refinement consists in exploiting the *temporal consistency* between consecutive frames to "transfer" labels across segmentations. The idea is that if a set of consecutive (in time) segmentations all mark a certain object as "interesting", then it is likely that they are correct; similarly, if no (or only few) segmentations include that object, it is probably safer to ignore it in the final output, especially if it is relatively isolated from other potential foreground objects. Two issues arise when trying to implement the above criterion: first, superclick segmentations are defined in terms of superpixel labels, and superpixel segmentation is not consistent in presence of motion; second, objects in a video typically move, thus the notion of "a certain object" across several frames implies the employment of a object/point tracking method.

Our approach addresses both issues: 1) we define a *temporal linking* between superclicks by extending the energy function, employed for superclick extraction, thus taking into account visual similarity between spatio-temporal regions; 2) we employ optical flow [48] to estimate where superpixels in frame t may have moved in frame $t + 1$: in practice, we introduce pairwise potentials on all pairs of superpixels $\{s_t, s_{t+1}\}$ such that s_t contains at least one pixel p_t whose projection $p_{t \rightarrow t+1}^{v_{p_t}} = p_t + v_{p_t}$ into frame $t+1$ under the motion vector v_{p_t} (i.e., v_{p_t} is the motion vector computed between frame t and frame $t + 1$ for location p_t) is part of superpixel s_{t+1} in frame $t + 1$. Of course, it is unlikely that each superpixel will appear in only one such link, which

allows to better "explore" the space around the estimated motion area, thus reducing the amount of error due to the optical flow and performing a more comprehensive analysis on the surrounding superpixels.

In the definition of the cost function employed for the temporal smoothing across frames $t-T$ and $t+T$, where t is the current processed frame and T is a constant which affects the number of frames involved in the temporal smoothing (i.e., $2T+1$), we assume to have identified superclicks for all the involved frames. In particular, we will refer to the same quantities as defined in Section 3.2 and add an apex relative to the frame they refer to: for example, l_s^t is the superclick label for superpixel s in frame t, S^{t+1} is the set of superpixels in frame $t+1$, and so on. The output label set will be identified by $\mathcal{L}$, and each label by l_s, without the temporal apex and they refer to the segmentation of the current processed frame. We can now introduce the energy function used for the final segmentation:

$$
\begin{aligned}
E_2(\mathcal{L}) = \sum_{\tau=t-T}^{t+T} & \left[\alpha_2 \sum_{s \in S^\tau} W_1(s, l_s, l_s^\tau) \right] + \\
+ \sum_{\tau=t-T}^{t+T} & \left[\sum_{(s_1,s_2) \in \mathcal{N}(S^\tau)} V_2(s_1, s_2, l_{s_1}, l_{s_2}) \right] + \quad (8) \\
+ \sum_{(s_1,s_2) \in \mathcal{N}_T(\cup_{\tau=t-T}^{t+T} S^\tau)} & V_2(s_1, s_2, l_{s_1}, l_{s_2})
\end{aligned}
$$

The first two lines of the cost function includes single-frame potentials, which consist of, respectively, unary potentials for each identified superpixel (first line) and pairwise potentials (second line) for each pair of superpixels belonging to the same frame. The last term (third line) enforces temporal smoothing, and consists of pairwise potentials computed over the set $\mathcal{N}_T(\cup_{\tau=t-T}^{t+T} S^\tau)$, which represents all pairs of superpixels (from all the frames in the considered time interval) satisfying the "temporal linking" described above, i.e., such that the two superpixels in each pair belong to temporally consecutive frames, and that at least one pixel belonging to one of them is projected onto the other by means of optical flow.

Similarly to V_1 (defined in Sect. 3.2), unary potential W_1 models whether superpixel s is more likely to be assigned to background or foreground *per se*. In this stage, we simply assign a constant value to the potential depending on whether it had been identified, at the previous stage (see Sect. 3.2), as a superclick or not (i.e., depending on l_s^τ). In detail, given superpixel s, we set corresponding *foreground cost* and *background cost*; the value of each cost depends on l_s^τ: if s was labeled as a superclick, we expect it to be more likely that it is foreground, so the background cost should be higher, and vice versa. W_1 is therefore computed as follows:

$$
W_1(s, l_s, l_s^\tau) = \begin{cases} \gamma_1 & \text{if } (l_s = 1 \wedge l_s^\tau = 1) \vee (l_s = 0 \wedge l_s^\tau = 0) \\ \gamma_2 & \text{otherwise} \end{cases}
$$
$$(9)$$

with $\gamma_1 < \gamma_2$.

Pairwise potential V_2 is defined as in Section 3.2, but in the last term (third line of Formula 8) of E_2 we employ it to evaluate the similarity not only between adjacent superpixels in the same frame, but also "temporally-adjacent"

(according to $\mathcal{N}_T$) superpixels in consecutive frames. In order to deal with errors in optical flow computation, we do not simply assign a constant based on the presence or absence of a temporal link between two superpixels in consecutive frames, but also verify that they are visually similar and in fact refer to the same object/region in both frames. Thus, we manage to enforce the criteria according to which superpixels overlaying the same region across different frames should all be assigned the same label.

Both E_1 and E_2 are binary pairwise energy functions with submodular pairwise potentials, and as such we minimize exactly them by graph-cuts in order to get the final segmentation for frame t. Some qualitative examples are shown in Fig. 5 (compared to those obtained by using only superclick extraction shown in Fig. 4): it is easy to notice the difference in terms of segmentation quality achieved by analyzing a single frame only and by employing temporal smoothing, which is able to extract much better objects' shapes. It should also be noted that most of the processing (e.g. superpixel extraction and optical flow) can be shared when processing frames one after another, thus reducing the main processing time to superpixel segmentation and computation of optical flow for a single frame only.

4 EXPERIMENTAL RESULTS

In this section we present the experimental results obtained by testing our gamification approach and link them to the state of the art on interactive video annotation and automated video object segmentation methods.

4.1 Datasets

For testing the accuracy of our method we created 7 game levels (each 300 frames long) from the following four datasets:

- Fish dataset [20]: we created two levels from the *ComplexBkg1* and *Standard1* video sequences, by choosing frame intervals with high density of ground truth frames and with several visible objects.
- I2R [49]: we created two levels from the *bootstrap* and *shopping_mall* videos, which are the most suitable ones for being used as game levels, since most videos in the dataset feature very few moving objects, in favor of dynamic background. The levels have been edited by selecting only parts with higher activity.
- ETH BIWI [50]: we created two levels from the *BAHNHOF* and *JELMOLI* sequences created for urban multi-person tracking. These videos were particularly interesting because of the large number of moving targets appearing at the same time; the disadvantage was that annotations were provided only at the bounding-box level, which we nevertheless treated as pixel-wise segmentations (which causes the reported performance to be lower than they actually are, as our method performs pixel-level segmentation).
- SegTrack v2 [51]: we created only one level by incorporating a subset of videos from the SegTrack v2 dataset (namely, *birdfall*, *bmx*, *cheetah*, *drift*, *hummingbird*, *monkey* and *monkeydog*) into a single sequence,
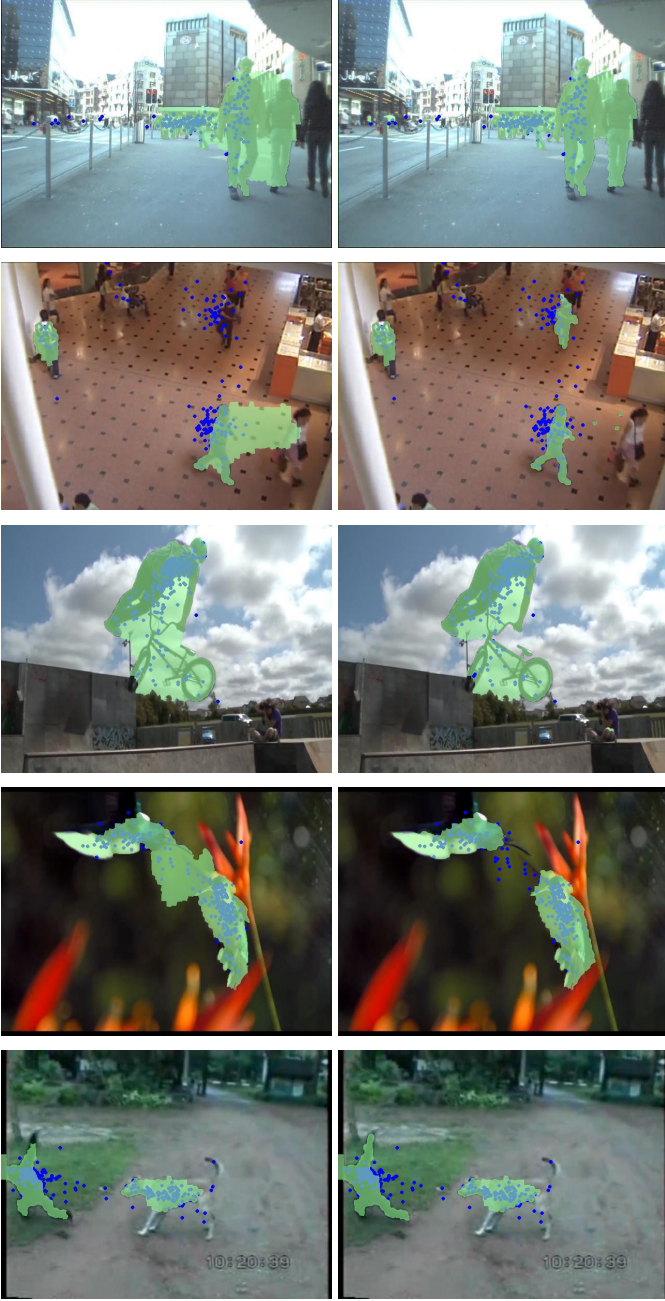
Fig. 5. Qualitative comparison between segmentations obtained when excluding (first column, see Fig. 4) and including temporal smoothing (second column).

since many of them were just few dozen frames long. We favored sequences with multiple objects, and excluded videos where the target, though moving, appeared at the same frame location due to camera motion (e.g. the *girl* sequence).

When we compared our methods to existing interactive video annotation methods and automated video object segmentation approaches we used the whole SegTrack v2.

4.2 Collected data

Our experiments involved only five players (two of them were children) outside of our research team, who were simply asked to compete with each other by achieving the highest possible score. The following information describes the amount of collected data and playing statistics:

- **Level time: 30 seconds.**
- **Game time (7 levels): 3:30 minutes.**
- **75 games played.**
- **Total play time: 9:18 hours.** On average, each participant played for 1:52 hours, which may seem a lot. In fact, in a real public gaming scenario the amount of collected data corresponds to having about 160 users playing just one game, which is easily achievable by publishing the game with very little advertising on a social network.
- **Total number of clicks: 235,799**; the average number of clicks per frame was 52.4.

4.3 Algorithm parameters

In Sect. 3, we introduced some parameters which control the trade-off between clicks and visual regularity in the segmentation process. We empirically set the values for those parameters, as follows: $\alpha_1 = 1/4$, $\alpha_2 = 1/5$, $U_s = 0.4$, $\beta_1 = 5$, $T = 2$, $\gamma_1 = 0.1$, $\gamma_2 = 0.9$ (W_1).

These parameter values were used to compute the results shown in the next section. It is important to note that the same parameters were used for all videos, although they have distinct differences in scenery, type of targets, motion patterns, motion speed, frame rate, etc. It is foreseeable that applying the same method to videos belonging to a more homogeneous set of videos would yield higher accuracy.

4.4 Segmentation results

This paper describes a interactive video object segmentation method based on spatio-temporal MRF optimization using the collaborative effort of multiple users. Thus, we first evaluated the role of temporal smoothing followed by the analysis of segmentation accuracy w.r.t to game play time, number of players, click delay and click quality. Then, we compared our method in terms of interaction times and segmentation accuracy with, respectively, recent state-of-the-art interactive video annotation methods and automated video object segmentation approaches. The metrics employed for performance analysis were pixel-level precision (Pr), recall (Rec) and F-measure (F_1) and average Pascal Overlap Measure (POM: intersection over union between T ground truth GT masks and output segmentations S) defined as:

$$\mathrm{Pr} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \tag{10}$$

$$\mathrm{Rec} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \tag{11}$$

$$\mathrm{F_1} = 2 \cdot \frac{\mathrm{Pr} \cdot \mathrm{Rec}}{\mathrm{Pr} + \mathrm{Rec}} \tag{12}$$

These performance metrics were computed by summing up the number of true positives, false positives and false negatives of each video category (i.e., without averaging across frames).

$$\mathrm{POM(GT, S)} = \frac{1}{T} \sum_{i=1}^{T} \frac{|GT_i \cap S_i|}{|GT_i \cup S_i|} \tag{13}$$

4.4.1 Role of spatio-temporal segmentation refinement

Fig. 5 shows a qualitative comparison in terms of segmentation outputs when employing only superclick extraction phase (see Sect. 3.2) and when exploiting the temporal consistency between consecutive frames of superclicks. Table 1 reports quantitatively how including spatio-temporal based refinement enhanced the segmentation accuracy. It can be noted that in some cases the accuracy gain was lower (ETH BIWI and I2R) than in others (Fish and SegTrack v2). This depends on the dynamics of the video sequences, i.e., in ETH BIWI and I2R the objects move slowly so users were able to click accurately on objects, while SegTrack v2 and Fish are characterized by strong camera and object motion resulting in much noisier input data, and the spatio-temporal based refinement proved effective to recover users' failures in identifying objects.

The lower performance achieved in ETH BIWI and I2R than the ones in Fish and SegTrack v2, instead, can be explained by the number of objects in the scene. Indeed, some ETH BIWI and I2R videos depicted very crowded scenes and given the limited number of users playing the game, not all objects present in the scene were accurately identified.

For all the following evaluations, we used the method including the spatio-temporal segmentation refinement.

4.4.2 Accuracy w.r.t. users' play time

Table 2 shows how segmentation results vary in relation to the amount of users' play time in terms of pixel-level precision, recall and F-measure.

It is easy to notice that, at the moment we interrupted the experiment, the accuracy had already started to slightly decrease as the participants played more games (i.e., as more clicks were being collected). Indeed, it is intuitive that as soon as enough clicks have been collected which allow to sufficiently highlight superclicks from the background, additional clicks are more likely to be noisy than to actually cover any missing foreground region. Of course, in a large-scale application scenario with hundreds or thousands of videos, the amount of play time needed to reach this "saturation point" is much higher and would require a very large user base and/or users' time.

4.4.3 Accuracy w.r.t. number of users

Among the advantages of exploiting gamification to solve a complex or large-scale task, one of the most important is the variety of data patterns contributed by different users of the system. This is especially important in multi-target tasks such as segmentation with multiple objects, as single users tend to be biased to select the same object across multiple games. We evaluated this tendency by comparing the segmentation accuracy obtained by taking into consideration the clicks from a single user (single-player scenario) with the accuracy obtained by the other four players (many-players scenario). The chosen user for the single-player scenario is the one who played most games (23); in order to compute the accuracy for the many-players scenario, we randomly sampled several sets of clicks such that each set had the same number of clicks per video as the chosen single player and averaged the results. Table 3 show the comparison between the two scenarios.

The comparison clearly shows that using clicks coming from many users yielded markedly better performance than having a single user gather the same amount of data. Of course, the reported performance is lower than the best accuracy in Table 2 because the click sets on which Table 3 was computed amount to less than 3 hours' cumulative play time.

To test the importance of integrating data from several participants in multi-target tasks, we compared the accuracy of the single-player and many-players with respect to the number of objects in a frame, across all videos. The results in Table 4 show that the difference in accuracy increases with the number of objects in a frame, hinting that, indeed, the variability introduced by a higher number of players positively affects a multi-target task with respect to fewer people working on it.

4.4.4 Accuracy w.r.t. click delay

While most algorithm parameters have a strictly mathematical meaning, click delay (i.e., the number of frames by which we shift user clicks to take human reaction delay into account) is particularly interesting to analyze, as it can be an important design choice which should be made by taking into consideration several factors such as user base, target speed, frame rate, etc. We evaluated segmentation accuracy for different click delay values, and the results are shown in Table 5. The specific value we used for our evaluations (2) is the one yielding the best average F-measure score, although the other values are quite close. It is interesting to see that the chosen value is not the optimal one for all videos – for example, it is not for the ETH BIWI and I2R videos: indeed, those are the video categories with the lowest dynamics, requiring fewer sudden reactions and thus making it easier for users to follow the objects.

4.4.5 Accuracy w.r.t. click quality

In order to measure the effectiveness of our click quality assessment approach, we evaluated segmentation accuracy using subsets of clicks belonging to different quality ranges (namely, [0–0.6[, [0.6–0.8[, [0.8–1]), as shown in Table 6. Unlike our tests concerning the variation of accuracy with respect to the number of players, we did not normalize the size of the click sets, as the three ranges were chosen so that they would approximately contain the same number of clicks. It can be seen that our measurement for click quality reflects the accuracy of the resulting segmentations; it should be noted that, as in Table 3, the reported results are sensibly lower than the best results (see Table 2) due to a lower number of used clicks. This demonstrates that click quality is necessary to achieve accurate segmentantions but the number of clicks is more important, i.e., it's more important to have many users than few high quality ones playing the game.

4.4.6 Comparison with the state of the art interactive video annotation methods

We also compared our method to existing interactive video segmentation ones [36], [41], [52] in terms of interaction times and accuracy. It has to be noted that for a single user the interaction time of our method depends only on

TABLE 1
Average segmentation accuracy for the video categories employed in our game in terms of precision, recall and F-measure, when we employ only superclick extraction (first row) and when we refine the output segmentation by means of spatio-temporal linking between superclicks in consecutive frames (second row).

Method	Fish			ETH BIWI			I2R			SegTrack v2		
	Pr	Rec	F_1	Pr	Rec	F_1	Pr	Rec	F_1	Pr	Rec	F_1
Superclick extraction	0.639	0.726	0.680	0.597	0.629	0.613	0.602	0.661	0.6304	0.673	0.763	0.716
Spatio-temporal refinement	0.684	0.821	0.746	0.647	0.636	0.642	0.606	0.696	0.6484	0.739	0.835	0.784

TABLE 2
Average segmentation accuracy for the video categories employed in our game in terms of precision, recall and F-measure, for different values of cumulative users' play time.

Playtime (hours)	Fish			ETH BIWI			I2R			SegTrack v2		
	Pr	Rec	F_1	Pr	Rec	F_1	Pr	Rec	F_1	Pr	Rec	F_1
9 h	0.684	0.821	0.746	0.647	0.636	0.641	0.606	0.696	0.648	0.739	0.835	0.784
8 h	0.711	0.814	0.759	0.632	0.670	0.650	0.609	0.720	0.660	0.721	0.827	0.770
7 h	0.735	0.787	0.760	0.672	0.612	0.640	0.663	0.670	0.666	0.770	0.819	0.794
6 h	0.738	0.760	0.749	0.655	0.644	0.649	0.681	0.677	0.679	0.758	0.796	0.776
5 h	0.707	0.592	0.644	0.689	0.564	0.620	0.756	0.571	0.650	0.816	0.758	0.786
3 h	0.780	0.372	0.504	0.716	0.472	0.569	0.812	0.368	0.507	0.883	0.617	0.727
1 h	0.869	0.149	0.254	0.816	0.339	0.479	0.909	0.228	0.365	0.929	0.450	0.607

TABLE 3
Average segmentation accuracy for the video categories employed in our game in terms of precision, recall and F-measure, for the single-player and the many-players scenarios.

Number of players	Fish			ETH BIWI			I2R			SegTrack v2		
	Pr	Rec	F_1	Pr	Rec	F_1	Pr	Rec	F_1	Pr	Rec	F_1
Single player	0.577	0.203	0.301	0.562	0.286	0.379	0.781	0.305	0.438	0.797	0.493	0.609
Multiple players	0.917	0.261	0.407	0.797	0.415	0.546	0.902	0.300	0.451	0.870	0.543	0.668

TABLE 4
Average segmentation accuracy in terms of precision, recall and F-measure, for the single-player and the many-players scenarios, with respect to the number of objects in a frame.

Number of players	1-2 objects			3-5 objects			6+ objects		
	Pr	Rec	F_1	Pr	Rec	F_1	Pr	Rec	F_1
Single player	0.798	0.458	0.582	0.564	0.219	0.315	0.687	0.200	0.310
Multiple players	0.868	0.502	0.636	0.760	0.285	0.415	0.852	0.290	0.432

TABLE 5
Average segmentation accuracy for the video categories employed in our game in terms of precision, recall and F-measure, for different click delay values.

Click delay	Fish			ETH BIWI			I2R			SegTrack v2			Average F_1
	Pr	Rec	F_1	Pr	Rec	F_1	Pr	Rec	F_1	Pr	Rec	F_1	
0	0.613	0.743	0.672	0.611	0.692	0.649	0.613	0.692	0.650	0.664	0.789	0.721	0.673
1	0.667	0.794	0.725	0.613	0.678	0.644	0.600	0.711	0.650	0.688	0.815	0.746	0.692
2	0.684	0.821	0.746	0.647	0.636	0.641	0.606	0.696	0.648	0.739	0.835	0.784	0.705
3	0.667	0.791	0.724	0.613	0.675	0.643	0.598	0.711	0.650	0.688	0.815	0.746	0.691

TABLE 6
Average segmentation accuracy for the video categories employed in our game in terms of precision, recall and F-measure, for different ranges of click quality. Along with each quality range, we report the fraction of clicks included in that range with respect to the total number of clicks.

Quality range	Fish			ETH BIWI			I2R			SegTrack v2		
	Pr	Rec	F_1	Pr	Rec	F_1	Pr	Rec	F_1	Pr	Rec	F_1
0.8 − 1 (32.7%)	0.817	0.423	0.558	0.777	0.477	0.591	0.872	0.391	0.540	0.896	0.633	0.742
0.6 − 0.8 (32.9%)	0.771	0.415	0.540	0.762	0.482	0.590	0.841	0.389	0.532	0.872	0.640	0.738
0 − 0.6 (34.4%)	0.392	0.223	0.284	0.370	0.267	0.310	0.407	0.319	0.358	0.493	0.335	0.399

TABLE 7
Comparison in terms of segmentation accuracy - measured as average POM in percentage, respectively, achieved within the first 50 secs of annotation (first row) and maximun value (with the related interaction times)- between our approach and other interactive video annotation methods on a subset of 10 frames extracted from SegTrack v2 video sequences

	[52]	[41]	[36]	Our method
POM within 50 secs	39.8	42.2	61.5	**72.4**
Maximum POM	56.6 (∼1,200 secs)	65.2 (∼500 secs)	84.3 (∼1,400 secs)	**72.4** (∼50 sec)

the video length, i.e., there is a linear dependency between number of annotated frames and annotation times, while existing interactive methods [36], [41], [52] show a non-linear (exponential-like) dependency. Nevertheless, using data generated by a single user in a single played game would result in poor accuracy performance (as discussed in the previous section), thus we consider, for comparison with the state of the art, the cumulative time spent by a single user in multiple game sessions (generally one frame is shown for 0.2 seconds in our game, thus if we select the clicks of one user in five game sessions, the interaction time would be of 1 second.). We selected randomly 10 frames from SegTrack V2 and we assessed how the segmentation accuracy changed w.r.t. to the interaction times. For our method, we considered the clicks of the user who played most games in several game sessions. Fig. 6 shows the interaction times of users against the achieved segmentation quality for our method and [36], [41], [52]. Our approach yielded a reasonably accuracy over the 10 considered frames after approximately 50 seconds of playtime (i.e., about 20 game sessions) while the other approaches needed much more time. Quantitatively, Table 7 reports the achieved segmentation accuracy (expressed by POM in percentage) over the 10 considered frames by all the comparing methods in two cases: a) within 50 seconds of annotation and b) the maximum achieved accuracy. Let us recall that the interaction time (50 seconds corresponding to about 20 game sessions) for our method is given by the playtime of a single user, and that the same value can be achieved by involving 20 users, in parallel, playing only one game session, i.e., with a very little human annotation effort as opposed to the existing other solutions [36], [41], [52].

4.4.7 Comparison with the state of the art automated video segmentation methods

Despite the proposed method is more inline with the research on interactive video annotation, it can be seen as a video object segmentation approach (with very little human intervention) and as such it is useful to link its performance with the state of the art on the automated methods. The comparison was again performed on the SegTrack v2 dataset (largely employed as a benchmarking dataset for video object segmentation), and we selected as comparing methods those ones posing the video object segmentation task as a superpixel labeling problem using spatio-temporal MRF optimization, namely, [1], [23], [24]. The results, in terms of average POM in percentage, are reported in Table 8.

Tables 7 and 8 indicate that our method performs better than automated video object segmentation methods and slightly worse than interactive video annotation approaches. This is not surprising since interactive video annotation

TABLE 8
Comparison in terms of segmentation accuracy - measured as average POM in percentage- between our approach and automated video object segmentation methods on SegTrack v2

	[23]	[24]	[1]	Our method
Average POM	53.5	59.3	64.4	**71.7**

tools require users to spend more time in providing accurate annotation but are barely usable in case of large video datasets. This makes our method a very good trade-off between accuracy and annotation times. However, when we used only points (uniformly taken from ground truth masks) within objects of interest (this is might be a typical interactive video annotation scenario where users might be asked to accurately select foreground pixels) we obtain a much higher accuracy with an average POM of about 0.85 as shown in Fig. 7.

Fig. 8 shows some failure cases of the proposed method: user clicks diverged substantially from objects' position hitting also background regions that were then classified as foreground. Our spatio-temporal refinement module was not able to recover such failures since in the previous phase (i.e., superclick extraction) several wrong superpixels were marked as superclicks.

4.4.8 Running times

Using $T = 2$, processing a single frame requires solving five MRFs for superclick extraction and one temporal MRF for accurate segmentation. Our Matlab implementation, run on a PC with a quad-core i7 CPU and 8 GB RAM, takes 3 seconds for superclick extraction in a single frame and 30 seconds for the temporal MRF (actually, for the four optical flow computations which link superpixels in time; MRF solving time is negligible), which would amount to 45 seconds in total. However, after the initial bootstrap phase, segmenting a new frame can benefit from already-extracted superpixels and computed the optical flow for previous frames, so processing is reduced to a single superpixel extraction and a single optical flow computation, resulting in a frame processing time of 10.5 seconds.

4.4.9 Available Resources

In the authors' webpage, the source code for object segmentation taking user clicks as input together with the videos used as game levels are available. We do not release the source code for the game, instead, we provide a web service where interested people can set up their game using their own videos and get the output segmentations as well as raw data (e.g., user clicks).
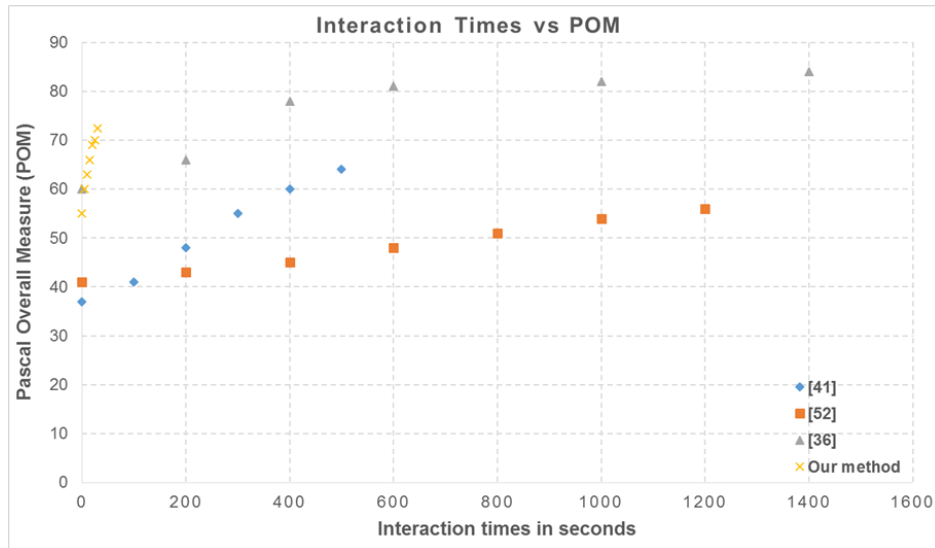
Fig. 6. Interaction times vs segmentation accuracy. The figure shows that with the proposed approach we get a fairly good segmentation quality just after 30 seconds. When we allowed users to spend more time on the annotation task, the method in [36] achieved the best performance with a POM of about 0.85 with an interaction time of about 1,400 seconds.
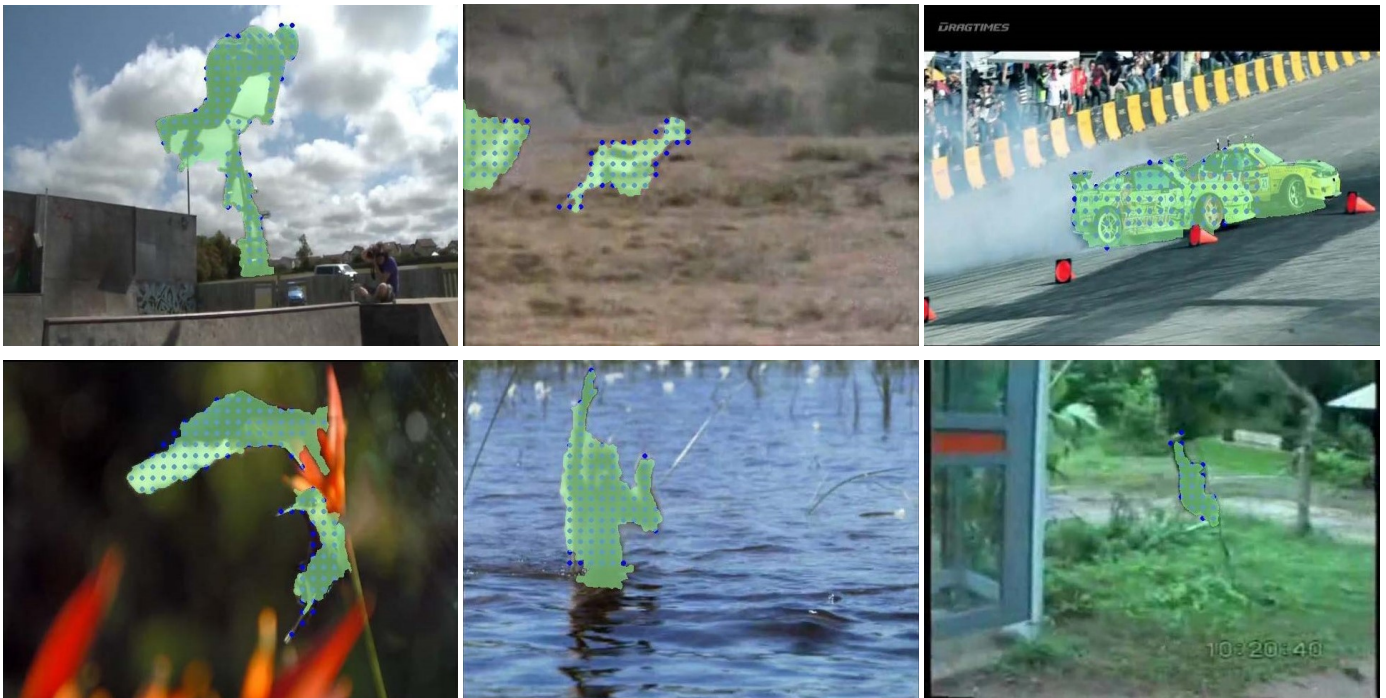

Fig. 7. Output segmentations when using only points within objects (i.e., taken from ground truth segmentation masks) of interest: blue dots are ground truth points while green regions show the yielded segmentation masks.

5 CONCLUDING REMARKS

In this paper we have described an interactive video object segmentation method which combines effectively games with a purpose strategy with collaborative human efforts. The performance analysis showed that our method outperforms in terms of segmentation accuracy state-of-the-art automated video object segmentation methods and is more suitable for large scale video annotation than classic interactive video annotation tools. We also demonstrated how including spatio-temporal regularization enhances greatly performance than using only spatial information and user-provided hard constraints. We also release the source code for human-guided video object segmentation as well as a web service that enables interested people to set up their game (with their own videos) and download user generated data. As future work, we plan to carry out experiments at a larger scale involving more users (as in [44]) and videos. In addition as inhibition of return and click delay are two important aspects of the approach, we plan to make them adaptive according to, respectively, play time (thus to enforce users to identify as many objects as possible) and video motion characteristics.

Fig. 8. Failure cases: User clicks were extremely inaccurate resulting in wrong object segmentations.

REFERENCES

[1] C. S. Daniela Giordano, Francesca Murabito, Simone Palazzo, Superpixel-based Video Object Segmentation using Perceptual Organization and Location Prior, in: Computer Vision and Pattern Recognition, 2015.

[2] B. Han, L. S. Davis, Density-Based Multifeature Background Subtraction with Support Vector Machine, IEEE Transactions on Pattern Analysis and Machine Intelligence 34 (5) (2012) 1017–1023. doi:10.1109/TPAMI.2011.243.

[3] P. Ochs, J. Malik, T. Brox, Segmentation of Moving Objects by Long Term Video Analysis, IEEE Transactions on Pattern Analysis and Machine Intelligence 36 (6) (2014) 1187–1200.

[4] X. Bai, J. Wang, G. Sapiro, Dynamic Color Flow: A Motion-Adaptive Color Model for Object Segmentation in Video, in: ECCV 2010, 2010, pp. 617–630.

[5] D. Zhang, O. Javed, M. Shah, Video Object Segmentation through Spatially Accurate and Temporally Dense Extraction of Primary Object Regions, 2013 IEEE Conference on Computer Vision and Pattern Recognition 0 (2013) 628–635.

[6] Y. J. Lee, J. Kim, K. Grauman, Key-segments for Video Object Segmentation, in: Proceedings of the 2011 International Conference on Computer Vision, ICCV '11, 2011, pp. 1995–2002.

[7] P. Ochs, T. Brox, Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions, Iccv (2011) 1583–1590.

[8] T. Brox, J. Malik, Object Segmentation by Long Term Analysis of Point Trajectories, in: ECCV 2010, 2010, pp. 282–295.

[9] K. Fragkiadaki, P. Arbelaez, P. Felsen, J. Malik, Learning to segment moving objects in videos, in: CVPR, 2015.

[10] L. von Ahn, L. Dabbish, Designing Games with a Purpose, Commun. ACM 51 (8) (2008) 58–67. doi:10.1145/1378704.1378719.

[11] D. Morrison, S. Marchand-Maillet, É. Bruno, TagCaptcha: Annotating Images with CAPTCHAs, in: Proceedings of the ACM SIGKDD Workshop on Human Computation, HCOMP '09, ACM, New York, NY, USA, 2009, pp. 44–45.

[12] S. Hacker, L. von Ahn, Matchin: Eliciting User Preferences with an Online Game, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '09, ACM, New York, NY, USA, 2009, pp. 1207–1216.

[13] L. von Ahn, L. Dabbish, Labeling images with a computer game, in: ACM Conference on Human Factors in Computing Systems, 2004, pp. 319 – 326.

[14] L. von Ahn, S. Ginosar, M. Kedia, R. Liu, M. Blum, Improving Accessibility of the Web with a Computer Game, in: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI '06, ACM, New York, NY, USA, 2006, pp. 79–82.

[15] L. Von Ahn, R. Liu, M. Blum, Peekaboom: a game for locating objects in images, Most pages (2006) 55–64. doi:10.1145/1124772.1124782.

[16] M. Addis, L. Boch, W. Allasia, F. Gallo, W. Bailer, R. Wright, 100 Million Hours of Audiovisual Content: Digital Preservation and Access in the PrestoPRIME Project, in: First International Digital Preservation Interoperability Framework (DPIF) Symposium, ACM, 2010.

[17] K. Siorpaes, M. Hepp, OntoGame: Weaving the Semantic Web by Online Games, in: S. Bechhofer, M. Hauswirth, J. Hoffmann, M. Koubarakis (Eds.), The Semantic Web: Research and Applications, Vol. 5021 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 751–766.

[18] S. Palazzo, C. Spampinato, D. Giordano, F. Murabito, Using the eyes to "see" the objects, in: ACMMM, 2015.

[19] O. Barnich, M. Van Droogenbroeck, ViBe: a universal background subtraction algorithm for video sequences., IEEE Transactions on Image processing 20 (6) (2011) 1709–1724. doi:10.1109/TIP.2010.2101613.

[20] C. Spampinato, S. Palazzo, I. Kavasidis, A texton-based kernel density estimation approach for background modeling under extreme conditions, Computer Vision and Image Understanding 122 (2014) 74–83.

[21] S. Liao, G. Zhao, V. Kellokumpu, M. Pietikainen, S. Z. Li, Modeling pixel process with scale invariant local patterns for background subtraction in complex scenes, in: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2010, pp. 1301–1306.

[22] B. Zhang, Y. Gao, S. Zhao, B. Zhong, Kernel Similarity Modeling of Texture Pattern Flow for Motion Detection in Complex Background, IEEE Transactions on Circuits and Systems for Video Technology 21 (1) (2011) 29–38.

[23] A. Papazoglou, V. Ferrari, Fast object segmentation in unconstrained video, Proceedings of the IEEE International Conference on Computer Vision (2013) 1777–1784.

[24] J. Lim, B. Han, Generalized Background Subtraction Using Superpixels with Label Integrated Motion Estimation, in: ECCV 2014, 2014, pp. 173–187.

[25] J. Deng, J. Krause, L. Fei-Fei, Fine-grained crowdsourcing for fine-grained recognition, Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2013) 580–587 doi:10.1109/CVPR.2013.81.

[26] S. Maji, Discovering a lexicon of parts and attributes, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7585 LNCS (PART 3) (2012) 21–30.

[27] A. Sorokin, D. Forsyth, Utility data annotaton with Amazon Mechanical Turk, Proceedings of the 1st IEEE Workshop on Internet Vision at CVPR 08 (c) (2008) 1–8.

[28] D. Parikh, D. Crandall, K. Grauman, Discovering localized attributes for fine-grained recognition, 2012 IEEE Conference on Computer Vision and Pattern Recognition (2012) 3474–3481.

[29] A. Parkash, D. Parikh, Attributes for classifier feedback, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 7574 LNCS (PART 3) (2012) 354–368.

[30] D. Parikh, C. L. Zitnick, Human-Debugging of Machines, Neural Information Processing Systems (2011) 1–5.

[31] S. Vijayanarasimhan, K. Grauman, Large-Scale Live Active Learning: Training Object Detectors with Crawled Data and Crowds, International Journal of Computer Vision 108 (1-2) (2014) 97–114.

[32] A. Salvador, A. Carlier, X. Giro-i Nieto, O. Marques, V. Charvillat, Crowdsourced object segmentation with a game, Proceedings of the 2nd ACM international workshop on Crowdsourcing for multimedia - CrowdMM '13 (2013) 15–20.

[33] V. Badrinarayanan, F. Galasso, R. Cipolla, Label propagation in video sequences, 2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (2010) 3265–3272.

[34] A. Fathi, M. F. Balcan, X. Ren, J. M. Rehg, Combining Self Training and Active Learning for Video Segmentation, Procedings of the British Machine Vision Conference 2011 (2011) 78.1–78.11 doi:10.5244/C.25.78.

[35] I. Budvytis, V. Badrinarayanan, R. Cipolla, Semi-supervised video segmentation using tree structured graphical models 35 (11) (2011) 2257–2264.

[36] N. S. Nagaraja, F. Schmidt, T. Brox, Video Segmentation with Just a Few Strokes, in: IEEE International Conference on Computer Vision (ICCV), 2015.

[37] B. L. Price, B. S. Morse, S. Cohen, LIVEcut: Learning-based interactive video segmentation by evaluation of multiple propagated cues, IEEE 12th International Conference on Computer Vision (ICCV) (Iccv) (2009) 779–786.

[38] V. Badrinarayanan, I. Budvytis, R. Cipolla, Mixture of Trees Probabilistic Graphical Model for Video Segmentation, International Journal of Computer Vision (2013) 1–16.

[39] Y. Li, J. Sun, H.-Y. Shum, Video object cut and paste, ACM Transactions on Graphics 24 (3) (2005) 595.

[40] C. Rother, V. Kolmogorov, A. Blake, "GrabCut": interactive foreground extraction using iterated graph cuts, ACM Trans. Graph. 23 (3) (2004) 309–314.

[41] S. Jain, K. Grauman, Supervoxel-Consistent Foreground Propagation in Video, in: D. Fleet, T. Pajdla, B. Schiele, T. Tuytelaars (Eds.), Computer Vision ECCV 2014, Vol. 8692 of Lecture Notes in Computer Science, Springer International Publishing, 2014, pp. 656–671.

[42] J. Donahue, K. Grauman, Annotator rationales for visual recognition, 2011 International Conference on Computer Vision (2011) 1395–1402.

[43] A. Vedaldi, S. Mahendran, S. Tsogkas, S. Maji, R. Girshick, J. Kannala, E. Rahtu, I. Kokkinos, M. B. Blaschko, D. Weiss, B. Taskar, K. Simonyan, N. Saphra, S. Mohamed, Understanding Objects in Detail with Fine-Grained Attributes, in: 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 3622–3629.

[44] I. Kavasidis, C. Spampinato, D. Giordano, Generation of Ground Truth for Object Detection While Playing an Online Game: Productive Gaming or Recreational Working?, in: Computer Vision and Pattern Recognition Workshops (CVPRW), 2013 IEEE Conference on, 2013, pp. 694–699.

[45] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Susstrunk, SLIC Superpixels, EPFL Technical Report 149300 (June) (2010) 15. doi:10.1109/TPAMI.2012.120.

[46] A. Jain, R. Bansal, A. Kumar, K. D. Singh, A comparative study of visual and auditory reaction times on the basis of gender and physical activity levels of medical first year students, Int J Appl Basic Med Res 5 (2) (2015) 124–127.

[47] V. Kolmogorov, R. Zabih, What Energy Functions Can Be Minimized via Graph Cuts?, IEEE Transactions on Pattern Analysis and Machine Intelligence 26 (2) (2004) 147–159.

[48] C. Liu, W. Adviser-Freeman, E. Adviser-Adelson, Beyond pixels: exploring new representations and applications for motion analysis, Proceedings of the 10th European Conference on Computer Vision: Part III (2009) 28–42.

[49] L. Li, W. Huang, I. Y. H. Gu, Q. Tian, Foreground object detection from videos containing complex background, Proceedings of the eleventh ACM international conference on Multimedia MULTIMEDIA 03 03 (2003) 2. doi:10.1145/957013.957017.

[50] A. Ess, B. Leibe, K. Schindler, L. Van Gool, A mobile vision system for robust multi-person tracking, Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on (2008) 1–8.

[51] F. Li, T. Kim, A. Humayun, D. Tsai, J. M. Rehg, Video segmentation by tracking many figure-ground segments, Proceedings of the IEEE International Conference on Computer Vision (2013) 2192–2199.

[52] L. Gorelick, F. R. Schmidt, Y. Boykov, Fast Trust Region for Segmentation, in: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), 2013.