VabCut: A Video Extension of GrabCut for Unsupervised Video Foreground Object Segmentation

Sebastien Poullot[1,2] and Shin'Ichi Satoh[1,3]
[1]*NII, Tokyo, Japan*
[2]*JFLI, CNRS, Tokyo, Japan*
[3]*University of Tokyo, Tokyo, Japan*

Keywords: Video Segmentation, Points of Interest, Outliers Removal, Frame Alignment, Motion M-layer, GrabCut.

Abstract: This paper introduces VabCut, a video extension of GrabCut, an original unsupervised solution to tackle the video foreground object segmentation task. Vabcut works on an extension of the RGB colour domain to RGBM, where M is the motion. It requires a prior step: the computation of the motion layer (M-layer) of the frame to segment. In order to compute this layer we propose to intersect the frame to segment with N temporally close aligned frames. This paper also introduces a new iterative and collaborative method for an optimal frame alignment, based on points of interest and RANSAC, which automatically discards outliers and refines the homographies in turns. The whole method is fully automatic and can handle standard video, i.e. not professional, shaky, blurry or else. We tested VabCut on the SegTrack 2011 benchmark, and demonstrated its effectiveness, it especially outperforms the state of the art methods while being faster.

1 INTRODUCTION

A clean segmentation of the objects in images and videos is a keystone to visual comprehension. For humans as for machines, for interpreting a scene, it is essential to be able to represent the relations and interactions between its macro components, i.e. some people and/or some objects, in their environment. The application range of video segmentation is wide, from smart video indexing to robot vision. This paper focuses on the separation of foreground objects from background in videos, i.e. video foreground object segmentation. The task is essentially a complex binary classification problem (foreground or background) of each pixel in a large spatio-temporal space. The human intervention is known to reduce the computational burden (Chockalingam et al., 2009; Brendel and Todorovic, 2009) whereas fully automated methods are known to be very computationally heavy (Lee et al., 2011; Ma and Latecki, 2012; Zhang et al., 2013).

Concerning image segmentation, ten years ago, the semi-automatic method GrabCut (Rother et al., 2004) has been a great step in the field. The user draws a rectangular bounding box around the object of interest then the algorithm automatically and quickly segments it. The method showed very accurate results and is still very popular. A weakness of the approach concerns the importance of the input: the bounding box drawn around the object must not contain any part of the object to segment, otherwise the algorithm is disrupted and may fail.

The core of our proposition is an extension of grabcut, namely VabCut, which takes as input the still image RGB layer and a motion M-layer for performing a fully automatic video foreground object segmentation. This M-layer is computed between N temporary close video frames. Compared to the state of the arts methods we claim that our approach has many practical advantages: it does not need any prior knowledge, learning or preprocessing of the video, it is fully automatic and fast. Furthermore it can handle non professional videos where low quality, bad focus, shaky hands, fast camera moves and so on may occur. Only two assumptions are done: **1.** a foreground object moves and **2.** it appears in temporally close frames (within 1/5 second). If the object stops moving the method can not properly segment it, if the object gets out of the video frame, it may take 1/5 second after new appearance to properly segment it again.

The next section presents some related works on the video segmentation. Our approach is developed in section 3 in three subsections; the first one presents an original method for optimal frame alignment, the

DOI: 10.5220/0004677103620371
In *Proceedings of the 9th International Conference on Computer Vision Theory and Applications* (VISAPP-2014), pages 362-371
ISBN: 978-989-758-004-8

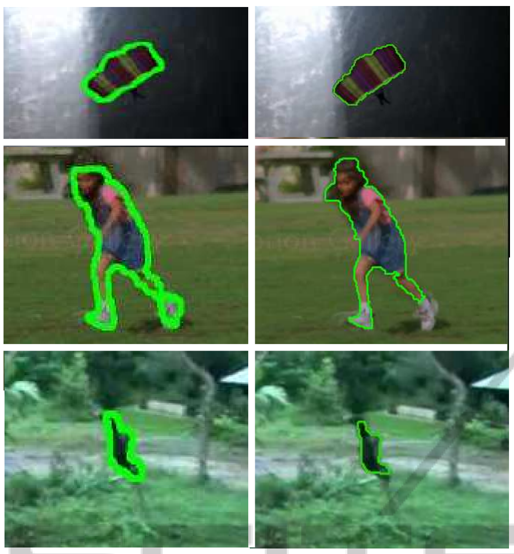
Figure 1: Some illustrations of our results (on right) compare to (Lee et al., 2011) automatic method on SegTrack2011. **Ours is almost 100 times faster**.

second one presents the M-layer creation, the third one presents the VabCut algorithm itself. Section 4 furnishes and discusses the parameters, then the quality and the efficiency are evaluated and compared to the state of the art methods on a reference foreground video object segmentation benchmark: SegTrack2011 (Tsai et al., 2010). The method shows good accuracy while fast processing. Moreover, we also contribute to the community by presenting an extension of the benchmark. Indeed, on two videos, not only one but two foreground objects appear, we manually annotated them and made them available for the community. We also furnish the results of our method on this new dataset.

2 RELATED WORKS

Various approaches have been proposed for video foreground object segmentation. The semi-unsupervised ones need some human intervention for initialization (Chockalingam et al., 2009; Brendel and Todorovic, 2009; Tsai et al., 2010), most of the time the contour of the objects to segment on the first frame of a shot. The initialization gives a very strong input to the method and is very useful to reduce the computational costs, usually 1fps can be processed. But if the video shots are very short, or the objects often come in and out, very frequent updates of the initialization is required and thus impractical.

Some other methods are fully automatic. (Grundmann et al., 2010) shows very impressive results on video segmentation on a large range of videos, camera motions, motions, objects, animals and people. It introduces a quite simple but smart hierarchical 3D spatial temporal segmentation. It cleverly segments videos in consistent volumes but can not clearly separate background and foreground objects, and a foreground object is generally divided in sub areas. The method requires quite heavy computations and memory, but a parallel out-of-the-core implementation can process 1 fps.

Recently, features trajectory based methods have shown great results, for example (Thomas and Jitendra, 2010; Ochs and Brox, 2011) propose to gather similar dense point trajectories in order to segment the frames. Unfortunately these approaches need a long term study of the features, i.e. can not be done online, and have quite an heavy computational burden.

From the image figure segmentation field, some notable works on co-segmentation (Joulin et al., 2012) could be an interesting approach for video foreground object segmentation cases. Each instance of a foreground object could be seen as an element of a low entropy class. One issue is that the background in a video does not change so much and could disrupt such method. Furthermore, for still images the methods are computationally expensive, it could be even more if adapted to the video case.

Recently, for video foreground object segmentation (Lee et al., 2011) have proposed a fully unsupervised method based on ”key-segments”. They are persistent edges that have a dynamic comportment different than neighbourhood. These edges are likely to belong to the foreground objects. The idea is to evaluate the ”objectness” of small areas conjointly in all frames, and group them as objects (by spectral clustering). The results are comparable to methods which require human intervention however it requires to analyse extensively the frame contents, i.e., a large part of the spatio-temporal space is explored. The method requires to pre-process the whole video at first and due to the complexity it is quite slow (5 minutes for a frame) and consumes a huge amount of memory. Despite the slow processing, the method shows better results than both pre-existing fully automatic and semi automatic methods on SegTrack2011 challenging benchmark (Tsai et al., 2010). (Ma and Latecki, 2012) worked on a some similar approach but with stronger constraints. One of the constraints is that a foreground object must appear in every frames of a video, which is quite a strong assumption. This constraint is respected in SegTrack2011 benchmark, their results shows some quality and speed improvements compared to (Lee et al., 2011). (Zhang et al., 2013) further explores this direction, they also exploit the optical flow to predict the new shape and position of the object in adjacent frames in order to shrink the space of objects to explore. The objects are inserted in a layered directed acyclic graph, where the longest paths are the ones presenting the more ”objectness”. To our knowledge their proposition shows the best score on SegTrack2011.

The aforementioned methods do not need to align the frames, they suppose that the frame rate is fast enough to consider that the object moves smoothly, i.e. its position and shape are very similar in next frames. Some other approaches relies on the frame alignment, once aligned their subtraction highlights the areas of motion, these areas potentially are foreground objects. The main advantage of such approach is to reduce the number of candidate areas for ”objectness”. (Sole et al., 2007) and (Ghanem et al., 2012) address this problem in sports videos where camera moves are quite smooth and typical. (Kong et al., 2010) aligns two video sequences in order to detect suspect objects in streets in videos shot from a moving car. Closer to our works (Granados et al., 2012) propose some background in-painting, i.e., removing a foreground object in a shot, based on multiple frame alignment. Each frame is decomposed in several planes, in order to process multiple homographies with the next and previous frames. The quality of their results are very impressive. The main drawback of the frame alignment approach is that if it is not accurate some false positive and loud noise occur and disrupt the algorithms.

3 VabCut:A VIDEO MOTION EXTENSION OF GrabCut

The idea of VabCut is simple, whereas GrabCut works on the RGB space for segmenting still images, we propose a solution that conjointly works on the motion information for segmenting video objects. The state of the art methods mostly separate image and motion, VabCut set them on a same level for segmentation: the motion is considered as an extra colour.

The approach consists of two main steps: **1.** compute the motion M-layer of the frame F_t using temporally close frames, **2.** perform VabCut algorithm with the RGB layer and the M-layer of F_t as inputs.

For computing the M-layer, different approaches can be used, basically direct registration or point registration methods. We here propose an original fast but robust point method frame alignment solution.

3.1 Robust Frame Alignment

The frame alignment is based on an estimation of the camera motion between two frames. Two main approaches for alignment can be distinguished: direct registration or point based registration. The first approach gives more precise correspondences, but is usually computationally very heavy (Bergen et al., 1992), the second one is faster but usually more approximative. We here propose an original point based approach which aims to keep the computation burden low while tackling some issues of the simple points matching approach.

A simple point matching process happens in 3 steps: **1.** points of interest detection and descriptor computation, **2.** descriptor matching, **3.** outliers removal and homography computation.

Concerning the points of interest and descriptors, during the ten last years, the computer vision community has been very influenced by David Lowe’s works, especially the well-known SIFT descriptors ((Lowe, 2004; Brown and Lowe, 2007)). This descriptor has shown some great capabilities for image matching even under some strong transformations. Since then the family of points of interest (PoI) and descriptors have been flourishing (SURF, FAST, BRIEF, ROOT-SIFT, etc). For temporally close image matching, we have 2 concerns, *first*, any PoI detector and any descriptor has some weakness (not representative, not discriminative, not robust, etc) depending on the visual contents of the images, *second*, the objects of interest in a video may attract the PoI detector while the background is not well described.

The aforementioned third step of a matching process is typically performed by RANSAC family algorithms, which are known to be robust to noise. The result of a RANSAC processing is an homography matrix that aligns the two frames. However if the number of outliers is too high compared to inliers the process fails. For example, it may mistakenly align foreground objects, suggesting that background has moved and the foreground objects have not.

To prevent this failure we propose an iterative and collaborative process that identifies and remove the outliers even when over represented in order to refine the frame alignment. In a nutshell, the process is as follows: **1.** compute and match multiple sets of PoI and local descriptors, **2.** for each set run RANSAC algorithm for removing some outliers and compute an homography between the two frames, **3.** evaluate the quality of each resulting homography, **4.** take the worst homography has a marker of outliers, remove the potential outliers from each sets, next iteration from RANSAC step. Each step is developed in the following of this subsection.

First, for a set of PoI and descriptor, (for example, DoG+SIFT), the features are extracted on frame F_t and $F_{t+\delta t}$. The local descriptor matching is performed between the two frames in a dual constraint way, let $D^s_{l_2}$ be the l_2-distance in the description space, a match between the descriptors $d_{t,i}$ and $d_{t+\delta t,j}$ is kept if:

$$\forall d_{t,k} \in F_t, D^s_{l_2}(d_{t,k}, d_{t+\delta t,j}) > D^s_{l_2}(d_{t,i}, d_{t+\delta t,j}) \wedge$$
$$\forall d_{t+\delta t,k} \in F_{t+\delta}, D^s_{l_2}(d_{t,i}, d_{t+\delta t,k}) > D^s_{l_2}(d_{t,i}, d_{t+\delta t,j})$$

Second, RANSAC is run on the two selected sets of points $\{P_t\}$ and $\{P_{t+\delta t}\}$ and returns the matrix $H_{t,t+\delta t}$ of a probable linear transformation (homography) between these sets. The point $p(x,y,1)$ in $F_{t+\delta t}$ is projected in F_t at position $p'(x',y',w')$, $p' = p \times H_{t,t+\delta t}$ where:

$$H_{t,t+\delta t} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \quad (1)$$

The Cartesian coordinates of point p' in frame F_t are $(x'/w', y'/w')$.

The two previous steps are run for multiple sets of PoI and local descriptors $S^i_{t,t+\delta t} = \{\{P^i_{t,t+\delta t}\}, \{D^i_{t,t+\delta t}\}\}$. For example the classic {DoG, SIFT} set, but also {SURF, SURF}, or {DoG, BRIEF}, etc. For each set $S^i_{t,t+\delta t}$ the homography $H^i_{t,t+\delta t}$ is computed.

Third, the homographies must be evaluated in order to select the most reliable one for alignment of the frames. The matching of two sets of 2D points between successive frames reduces the case to an affine transformation, therefore, in homography matrix 3.1, $h_{31} = h_{32} = 0$ and $h_{33} = 1$. h_{13} and h_{23} correspond to the translations, respectively T_x and T_y, the typical camera moves for following a foreground object. They are the consequences of the rotations on camera vertical and horizontal axis. h_{11}, h_{12}, h_{21} and h_{22} are the consequences of multiple factors. Mostly it depends on the rotation $R^\theta_{t,t+\delta t}$ of angle θ of the camera on its focal axis, and on the shearing $Sh_{t,t+\delta t}$, deformation due to the concavity of the lens, and forward, backward moves of the camera:

$$\begin{pmatrix} h_{11} & h_{12} \\ h_{21} & h_{22} \end{pmatrix} \sim \begin{pmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{pmatrix} \begin{pmatrix} sh_x & 0 \\ 0 & sh_y \end{pmatrix}$$

The shearing depends on positions $\{P_t\}$ and $\{P_{t+\delta t}\}$ in their respective frame and on the camera translation $Tc_{x,y,z}$. The pixels on the borders undergo a stronger shearing. If the PoI are considered to appear uniformly on the frames, each set of point $\{P_t\}$ undergo an identical shearing. Therefore, for scoring an homography $H^i_{t,t+\delta t}$, $Sh^i_{t,t+\delta t}$ can be neglected, and h_{11}, h_{12}, h_{21}, h_{22} are only consequences of the rotation of the camera on its focal axis $R^\theta_{t,t+\delta t}$. Contrary to the translations, this shooting move is not natural and should not be strong. If it is strong it is usually the sign of an alignment failure, not of a bad camera move, therefore it should be penalized. For a set of PoI and descriptors C_i, we propose to define an alignment cost as follows:

$$S_a(H^i_{t,t+\delta t}) = \sqrt{(h^i_{11}-1)^2 + (h^i_{22}-1)^2 + {h^i_{21}}^2 + {h^i_{12}}^2}$$

If the camera does not rotate $h^i_{11} = h^i_{22} = 1$ and $h^i_{12} = h^i_{21} = 0$, thus $S_a(C^i_{t,t+\delta t}) = 0$. Between two camera moves estimation, the one depicting the more realistic camera move is kept: for n set $S^i_{t,t+\delta t}$ of PoI and descriptors, the homography H_i having the lower cost $S_a(H^i_{t,t+\delta t})$ is selected:

$$H_s = \arg\min_{i \in [0,n]} S_a(H^i_{t,t+\delta t})$$

An important point is that the score is designed to promote the camera natural moves. However the homography containing shearing and other undergone transformations consequences is fully applied for performing the alignment between frames.

Fourth, some spatial arrangements of the PoI in the frames induce bad frame alignment. Basically, if the number of PoI extracted from the background is small compared to the number of PoI extracted from the foreground objects, the process fails. Indeed, if such unbalanced distribution occurs the foreground objects points become the references for frames alignment. The resulting alignment leads to messy irrelevant segmentations. RANSAC is a non-deterministic algorithm and could be run many times on each set C_i until a low alignment cost is found. However if the sets of points contain spacial consistent noise (the object points), the probability is very low. To prevent this, we propose to iteratively remove points on the foreground objects in order to correct the balance between foreground and background. Instead of focusing on the best alignment, which are also possibly tricked by the foreground objects, we propose to investigate the worst alignment for improving the other ones.

Each alignment is based on a set $S^i_{t,t+\delta t}$ which point positions $P^i_{t,t+\delta t}$ may not be shared (for example points detected by DoG and SURF), however their proximity in the frame plane can be exploited. Our proposition is to consider the positions of the PoI of

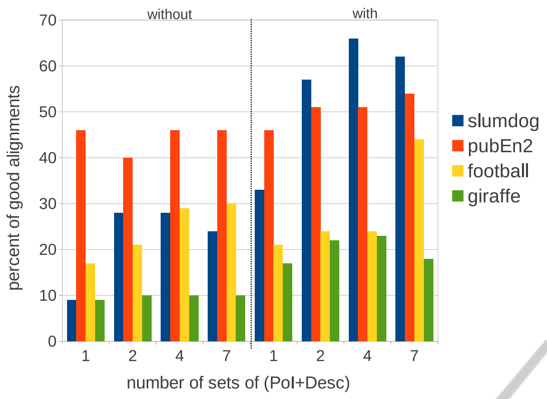

Figure 2: Correction of the alignments on 4 very challenging videos. On the left **without** collaborative iterative procedure, on the right **with** the procedure, both based on 1, 2, 4 or 7 sets of PoI/descriptors. The base line is the first column, it corresponds to a standard RANSAC alignment using DoG/SIFT set of features. The percentages of good alignment clearly increases with the collaborative iterative method.

the worst alignment as markers of the outliers: for each other set $S^i_{t,t+\delta t}$, the PoI close to these positions (under a threshold distance Th_D) are removed and the RANSAC algorithm is run again. The procedure stops if, at any time, an alignment cost is below the threshold Th_{S_a}. If this condition is not satisfied before all the PoI are removed from all sets $S^i_{t,t+\delta t}$, the best alignment seen along the iterative process is kept.

We propose here a short evaluation of our proposition for robust frame alignment. Figure 2 displays a hand made evaluation of the reliability of the homographies performed on three videos proposed by (Grundmann et al., 2010) and the giraffe video (also illustrated in figure 3). These videos were chosen because they are very inconvenient: they are close ups where averagely 80% of the descriptors are on the foreground objects in motion (95% for the giraffe). We counted the correct homographies frame to frame with the simple processing, four first columns group, and the collaborative iterative processing, four next ones. For each case, 1, 2, 3 or 7 sets of PoI and descriptors were computed. The simple process using one set can be considered as the baseline (first column). It can be seen that the collaborative iterative method using 7 sets (last column) strongly improves the reliability of the alignment. Overall it gives 125% more good alignments than the baseline. Figure 3 illustrates a typical case of successful correction.

Only one homography $H_{t,t+\delta t}$ is kept between two frames and considered as the right camera motion estimation. The frame $F_{t+\delta t}$ is projected on F_t plane. The resulting image $F'_{t+\delta t}$ and F_t can then be subtracted at pixel level in order to obtain a map that

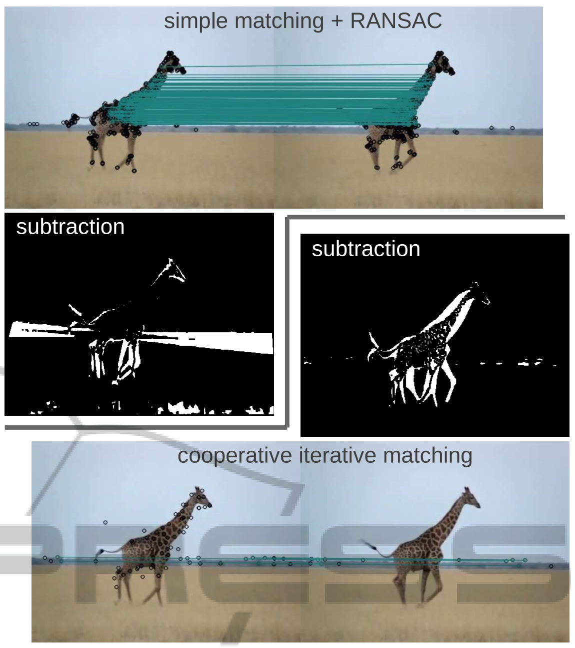

Figure 3: A successful correction case in the giraffe video. The result of a simple RANSAC processing is displayed on top left, most of the matches are found on the giraffe body. The alignment is wrong and the frame subtraction mask (in black and white, it is thresholded for comprehension) reveals the skyline, the giraffe legs and its head. On bottom right, the collaborative iterative process found a better alignment after two iterations. It can be seen that the selected matches are now only located on the skyline. The alignment and the resulting subtraction are satisfactory.

highlights the areas of motion, the M-layer.

3.2 Spatio-temporal M-layer

The subtraction of two frames F_t and $F_{t+\delta t}$ is informative but not precise about the spatio-temporal location of the objects (see figure 3) At least three frames must be processed to obtain a satisfactory localization of the object (it gives a kind of spatio-temporal triangulation). However, if an object in $F_{t-\delta t}$ comes back to the same position at $F_{t+\delta t}$, the triangulation of the three frames $F_{t-\delta t}$, F_t and $F_{t-\delta t}$ gives a result equal to the subtraction of $F_{t-\delta t}$ and F_t or F_t and $F_{t+\delta t}$. For tackling this issue, we propose to ensure the spatio-triangulation by using more than three frames. Figure 4 illustrates the proposition with five frames. The frame F_t is intersected with four other ones, two for the original triangulation, and two other ones for ensuring. To perform the intersection of this three temporary M-layers, the *min* operator at pixel-level is applied. The more temporary M-layers are used the less false positive pixels remain in the final M-layer, on the other hand some true positive pixels are lost as well.

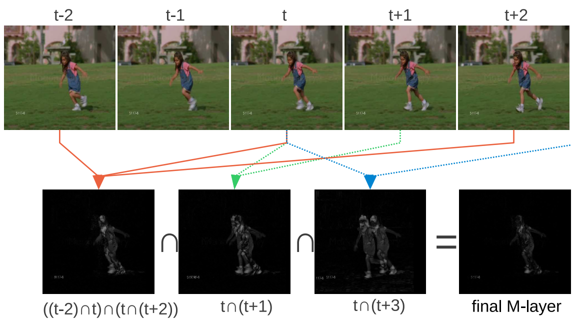

Figure 4: On top row, the successive frames of a video. On bottom row, the M-layers, using F_{t-2}, F_t and F_{t-2}, using F_t and F_{t+1}, and using F_t and F_{t+3}. The final M-layer is the intersection of the three previous one.

In the evaluation section, it will be seen that removing the false positive is essential for the method.

An advantage of this super triangulation is that the noise caused by the approximation of the alignment is partially removed. A drawback is that the absolute position of the object in F_t must be shared by the four other frames. If the object or/and the camera move too fast, this condition may not be fulfil and the process may fail.

3.3 Video Foreground Object Segmentation with VabCut

GrabCut (Rother et al., 2004) is widely used as a step for image and video segmentation (Chen et al., 2009; Yang et al., 2011). Originally it is an interactive method, the user intervention is required to draw a bounding box around the object to segment in a picture. Vabcut algorithm is fully automatic, the bounding box is drawn depending on the location of the areas of motion, it works on the RGBM domain, color plus motion, and optimizes the areas models (FG and BG) generated at initialization.

VabCut Bounding and Super Bounding Box. In order to automatically find a bounding box for VabCut, a binary version of the previously computed motion M-layer is computed. For this, an adaptive threshold based on local intensity Th_{li} is applied. The areas of activity are cliques of white pixels (white connected surfaces). The cliques close to each other, under a distance threshold Th_{db}, are merged as one blob. Each blob B_i is a potential foreground object. From each blob B_i a bounding box bb_i can be automatically drawn. Due to previous processings, a blob may be smaller than the real object, therefore its bounding box is enlarged by E_{bb} pixels in every direction.

Speaking about the original GrabCut algorithm, if it is performed using bb_i directly on the original frame, the part inside bb_i is labelled as unknown, the part outside is labelled as background. The algorithm is based on Gaussian mixture models (GMMs) of the colours. At initialization two models are build, one from the background area, one from the unknown area. When the background is large compared to the unknown area, they may share some visual similarity, as a result their initials GMMs are inextricable and GrabCut fails. If the background is different from the background contained in the bounding box Grabut fails as well. For these two reasons only a close area around the bounding box should be considered as background. Therefore for VabCut a super bounding box sbb_i is computed by enlarging the size of bb_i by a factor F_{sbb}. Then VabCut is run using the same bounding box bb_i and the area between sbb_i and bb_i as background.

Inside VabCut. In the following, we fit to the notations used in (Rother et al., 2004) so that the reader can easily refer to this article. In VabCut the pixel z_n at position (x,y) in frame F_i is defined in 4 dimensions:

$$z_n = (R_n, G_n, B_n, M_n) \tag{2}$$

where M is the value in the M-layer previously computed for this frame. The colorimetric distance between 2 pixels is a regular Euclidean L_2-distance using uniformly the 4 dimensions:

$$D_{L_2}(z_n, z_m) = \sqrt{(R_n - R_m)^2 + (G_n - G_m)^2 + (B_n - B_m)^2 + (M_n - M_m)^2} \tag{3}$$

The change of space has an impact on the whole algorithm. The distance is used for performing the K-means that initializes the GMMs models, and is also used in the Gibbs energy $\mathbf{E}$ computation of the segmentation:

$$\mathbf{E}(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) = U(\underline{\alpha}, \mathbf{k}, \underline{\theta}, \mathbf{z}) + V(\underline{\alpha}, \mathbf{z}) \tag{4}$$

where $\underline{\alpha}$ is the vector of areas the pixels belong to (here it can take two values, $\underline{\alpha}_n = \{FG, BG\}$), $\mathbf{k}$ is the vector of components of the GMM the pixels belong to ($\mathbf{k}_n \in [1, K]$), $\underline{\theta}$ is the vector of the models of area of the pixels ($\underline{\theta}_n = \{FG, BG\}$), $\mathbf{z}$ is the vector of pixels values in 4 dimensions (as in equation 3.3). U measures how the segmentation fit to the model and V is the smoothness term. The algorithm iteratively minimizes the energy $\mathbf{E}$. Please refer to (Rother et al., 2004) for more details on the algorithm.

Independent Optimization of the GMM of the BG and FG Areas. In the basic GrabCut algorithm, the number of Gaussian distribution in the mixtures is fixed to $K = 5$ for the background as for the foreground. However the compositions of these two areas can be very simple, in that case they are represented by too many Gaussian distributions. On the opposite, they can be very complex, and in that case they are not represented by enough Gaussian distributions. The relevancy of a model can be estimated by computing its entropy ξ, or the error between the prediction and the original data. A low entropy or a low error means that the model fits well to the data.

In the following we first propose to estimate the entropy of the background and foreground models in order to pick the right K for each one. The GrabCut algorithm is initialized by a K-means algorithm in order to build K Gaussian distributions for both background and foreground pixels. Originally the K-means algorithm iteratively refines the positions of K centroids in order to satisfy the lower possible entropy of the system. At each iteration step, this entropy is defined as the sum of the L2-distances between each data point S_i and its representative centres C_{S_i}.

$$\xi_{KM}(K) = \sum_{i=1}^{nbSamples} d_{L2}(S_i, C_{S_i}) \tag{5}$$

A low entropy characterizes a model in which the data points are closed to their representative centroid: the clusters are compact and the data well separated. Depending on the initialization of the representative centroids the results may vary a bit and it is hard to know whether the global minimal entropy as been found. Usually the algorithm stops after a predefined number of iterations or when the attribution of the data points to the representative centroids is stable between two iterations (the entropy is stable). By performing many initial random draws an optimal model of lowest entropy may be found, but considering the description space and the initial number K, the number of solutions $|S|$ to explore is way too large. Let note $|S_f|$, the feature space size, then $|S| \approx |S_f|^K$. In our practical case, for 4 colours channels on the $[0;255]$ range and $K = 5$ then $|S| \approx 1.3 \times 10^{48}$. Of course many solutions can be pruned out with sophisticated techniques (smart sampling for example), however performing random draws gives very few chances to improve the model for a huge loss of computations.

As mentioned before, the entropy of the model may remain high because K does not fit to the real number of observable Gaussian distributions in the data. Therefore we propose to run the K-means algorithm on a range of values. For each value the entropy of each area $\xi_{KM}(area, K)$ is estimated by the K-means entropy (equation 3.3). The K offering the lower entropy gives the optimal number of Gaussian distributions to use for VabCut algorithm. In order to have a data model as refined as possible, two independent models, one on the background pixels, one on the foreground pixels, are computed, having respectively K_{BG} and K_{FG} Gaussian distributions:

$$K_{BG} = \underset{K \in [K_{min}, K_{max}]}{\operatorname{argmin}} \xi_{KM}(BG, K) \tag{6}$$

$$K_{FG} = \underset{K \in [K_{min}, K_{max}]}{\operatorname{argmin}} \xi_{KM}(FG, K) \tag{7}$$

A second proposition is to evaluate the mixture of Gaussians generated by K-means using an Expectation Maximization (EM) classifier. Again, one classifier is used for each area. The train set and the test set are the same: all the pixels from each area. An EM classifier return a list of probability $P(z_n) = p_1(z_n), p_2(z_n), ..., p_K(z_n)$ for each pixel to belong to each model of a K-GMM. We define the relevancy of the model according to the EM classifier error of prediction:

$$\xi_{EM}(area, K) = \sum_{i=n}^{nbSamples} p_j(z_n),\ z_n \in K_j \tag{8}$$

The selected K_{BG} and K_{FG} respectively for the BG and the FG areas become:

$$K_{BG} = \underset{K \in [K_{min}, K_{max}]}{\operatorname{argmax}} \xi_{EM}(BG, K) \tag{9}$$

$$K_{FG} = \underset{K \in [K_{min}, K_{max}]}{\operatorname{argmax}} \xi_{EM}(FG, K) \tag{10}$$

The comparison between the two GMM optimizations are given in the next and last evaluation section.

4 EVALUATION

The tests are run on a laptop, processor Intel i7-3520M@2.90GHz, using only one core.

Parameter Consideration. For setting up the parameters we tested our algorithm on a large set of videos, composed of ten Youtube videos, all animals into the wild, the set from (Grundmann et al., 2010) and the segTrack2011 benchmark (Tsai et al., 2010). This one is also the base for comparison with the state of the art methods.

The frames can be more or less temporally distant, we tested our algorithms on 15fps to 30fps video,

Table 1: The top part of the tab shows the results on the ground truth furnished by (Tsai et al., 2010). Average mean pixel error comparison for SegTrack 2011, the lower the better. F_1 is indicated as well, the higher the better. The bottom part of the tab adds the results on the complete ground truth, manually annotated by our team.

Method	VabCut EM-5f		VabCut KM-5f		VabCut KM-3f		VabCut KM-OptFlow		GrabCut NO-3f	
parachute	156	**0.98**	**141**	**0.98**	155	**0.98**	281	0.96	152	**0.98**
girl	**577**	**0.95**	683	**0.95**	884	0.93	1291	0.89	1273	0.90
monkey	**748**	**0.70**	897	0.67	908	0.67	1195	0.57	764	0.67
deer	**861**	**0.70**	870	**0.70**	1019	0.65	1047	0.59	917	0.61
birdfall	215	0.75	210	0.76	**202**	**0.78**	402	0.64	214	0.74
total	**2557**		2801		3168		4216		3320	
monkey and dog	**1749**	**0.52**	1895	0.50	1850	**0.52**	2103	0.46	1820	0.47
cheetah and deer	1322	0.67	1298	0.68	**1255**	**0.69**	1721	0.54	1709	0.52
total all	**4019**		4227		4346		5789		5125	

in any case all the frames are used. It exactly corresponds to figure 4, the basic algorithm takes 3 frames (t-2,t,t+2), the full version takes 5 frames.

For the collaborative iterative frame alignement (subsection 3.1) 7 sets of PoI and local descriptors are computed, namely: DoG+SIFT, DoG+Brief, SURF+Brief, ORB+ORB, FAST+Brief, FAST+SIFT, MORPHE+Brief. They are all set with default parameters. MORPHE is a home made descriptor, it identifies singularity points (angles and straight lines) on the canny edges filtering of a frame. The quality threshold is $Th_{S_a} = 0.05$, the distance threshold is $Th_D = 10pix$ for low definition videos (segTrack2011 for example) and proportionally increases with the video definition (these two threshold appear in the point **fourth** of the subsection 3.1).

About the bounding box and super bounding box (subsection 3.3 first part), the intensity threshold Th_{li} at pixel level is adaptive to the local mean intensity (a linear function $(0,10) \rightarrow (255,30)$ computed on the 9×9 neighbour pixels). The threshold merging distance between blobs is fixed to $Th_{db} = 25pix$ for low definition videos (segTrack2011 for example) and proportionally increases with the video definition. $E_{bb} = 5pix$ and the surface of a sbb_i is set to $1.75 * surface(bb_i)$.

Finally, for VabCut GMM optimization (subsection 3.3 last part), $K \in [2-11]$ values are tested.

SegTrack 2011 Benchmark. Our approach was tested on the video database proposed by (Tsai et al., 2010). It is composed of 6 short videos, in which each frame has been hand-labelled. The ground truth of each frame is a mask containing the position of only one moving object in each sequence. The "penguins" video results are not given here, indeed it contains a colony of penguins, and only one is annotated as a good detection. Our algorithm tends to detect all the penguins. The scores are average per frame pixel error rate from (Tsai et al., 2010):

$$\varepsilon(S) = \frac{XOR(S,GT)}{F} = \frac{FP+FN}{F} \qquad (11)$$

with S is the resulting segmentation, GT the furnished ground truth and F the number of frames of the sequence. FP are the false positive pixels and FN are the false negative. We also indicate the F_1 scores:

$$F_1 = \frac{2 \times P \times R}{P+R} \qquad (12)$$

with P as precision and R as recall:

$$P = \frac{TP}{(TP+FP) \times F},\ R = \frac{TP}{(TP+FN) \times F} \qquad (13)$$

where TP are the true positive pixels. This measure expresses the balance and the completion of the segmentation.

The ground truth provided by (Tsai et al., 2010) is incomplete, on the video *cheetah* and *monkey*, not only one animal appears but two, therefore we manually extended the ground truth. First we compared the results on the original ground truth by incorporating incrementally our propositions, see table 1. The nomenclature for the method used is as follows: on the first line, GrabCut or Vabcut method, on the second line, 5f or 3f (5 or 3 frames intersection) or optical flow is the method for computing the M-layer, EM, KM or NO (EM, K-means or no optimization) is the method for optimizing the GMMs.

As can be seen, the VabCut algorithm dramatically improves the quality of the results on the *girl* video, while it is quite stable on the other videos. Improving the results on the parachute is very challenging (see F_1 scores), the object is very well separated from its background and the segmentation is

Table 2: Average mean pixel error comparison for SegTrack 2011, the lower the better, 1:(Zhang et al., 2013), 2: (Ma and Latecki, 2012), 3: (Lee et al., 2011)

Method	VabCut EM-5f	(1)	(2)	(3)
parachute	**156**	220	221	288
girl	**577**	1488	1698	1785
monkey	748	**365**	472	521
deer	861	**633**	806	905
birdfall	215	**155**	189	201
total	**2557**	2861	3377	3700
benefit		10.6%	24.3%	30.9%

very good even with the original GrabCut algorithm. The monkey case is quite similar, the contrast with the background is very strong and GrabCut can already perform quite good.

The combination VabCut-EM-5f gives the best overall results, which means that VabCut requires a M-layer with not much false positives in it, which fits to the original GrabCut property (and as mentioned in subsection 3.2). The results with VabCut-OptFlow confirms this property, the flow motion does not fit very precisely to the object contour, the M-layer contains more false positive motion, consequently the results are degraded.

About the new ground truth, an important point must be mentioned. The improvement on *monkey and dog* is quite low, indeed the dog appear in less than half of the frames and does not move so much. As VabCut is based on motion, it does not perform so good. On the other hand, the improvement on *cheetah and deer* is very high, here the two animals are in motion all along the video, VabCut brings some benefit. Figure 5 illustrates some results on two videos from the benchmark, 6 shows some more qualitative results on videos proposed by (Grundmann et al., 2010).

Compare to the state of the art automatic methods (table 2), our overall score is the best, especially because of the *girl* video. On the *deer* and *monkey* videos, it is harder to make a straight comparison, as our method tends to find the two animals in the two videos. Plus, we want to point out that for the *monkey* video, there are 10 frames is a row (from 212 to 222) where our method can not succeed. Indeed the monkey is almost out of the scope and run away, the resulting M-layers are void in this subsequence (this possible issue is mentioned at the end of subsection 3.2). If this subsequence is removed, the score becomes *646* instead of *748* (first column of tables 2 and 1).

About the time computation, our VabCut-EM-5f is quite slow, about 18 seconds per frame on the SegTrack2011 videos (320x240). By using only half of

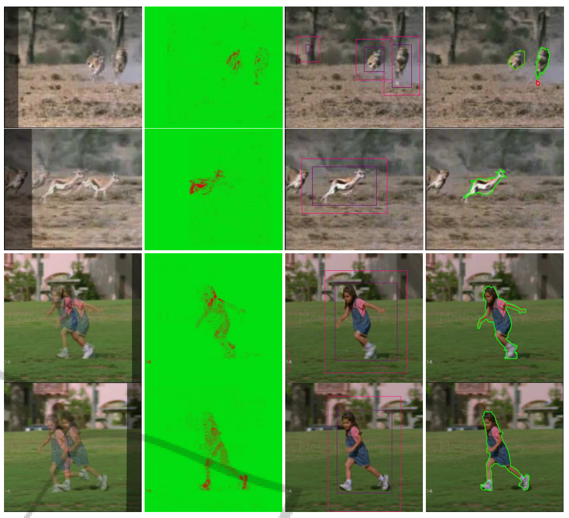
Figure 5: Some results on SegTrack 2011. The first column shows the superimposition of the frames after alignment (only 2 frames for clarity), the second one shows the M-layer (motion is in red), the third column shows the bounding boxes and super bounding boxes, the fourth one shows the final segmentation. The 2 tops rows are from the *deer and cheetah* video, on the top one, the 2 animals are found, on the bottom one, only the deer, indeed the cheetah is not in the overlapping part of the frames (see the M-layer). The 2 bottom rows are from the *girl* video, showing quite accurate segmentation.

the pixel for learning and the other one for predicting, the quality is stable and the computation time for a frame is lowered to 10 seconds. The VabCut-KM-5f is much more faster, requiring 4 seconds per frame. If the method is still far from real time, it is however faster than (Lee et al., 2011) method, which requires about 300 seconds per frame, (Ma and Latecki, 2012) is not clear about time consumption, (Zhang et al., 2013) does not mention it. The memory consumption is almost null, and the process can be performed on a streaming video with no length limit.

5 CONCLUSIONS AND FUTURE WORKS

In this paper, we propose an efficient and effective automatic one-pass method for video foreground object segmentation. It is based on a motion estimation between temporally close frames after point based alignment, and VabCut an extended version of GrabCut in the motion domain. The method has many practical advantages. It is fully automatic, does not require any preprocessing of the video, does not learn any models, and does not consume memory. It can handle everyday's life videos, shaky hands, fast camera or objects moves. On the challenging SegTrack2011 benchmark

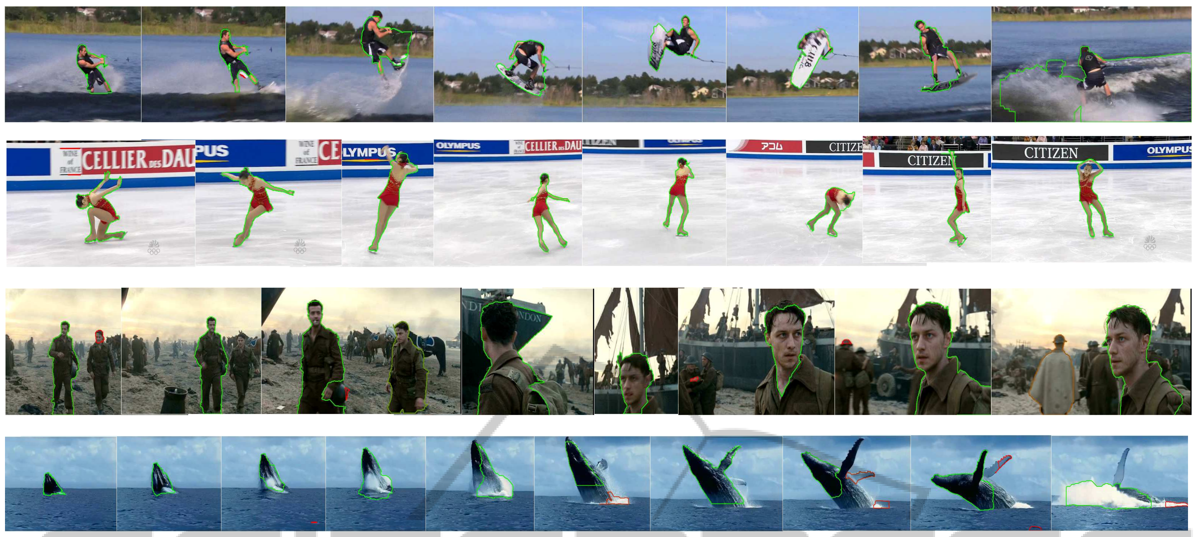

Figure 6: Some of our segmentation results on the videos proposed by (Grundmann et al., 2010).

it achieves better results than the state of the art automatic methods, which may also be very much slower. As limitations, the proposed method can not detect still foreground object, moreover if a foreground object stops moving it is lost. Our next works will focus on the consistency over time of detected foreground objects, in order to propagate, clean and track the objects.

REFERENCES

Bergen, J. R., Anandan, P., Hanna, K. J., and Hingorani, R. (1992). Hierarchical model-based motion estimation. In *ECCV*, pages 237–252.

Brendel, W. and Todorovic, S. (2009). Video object segmentation by tracking regions. In *ICCV*.

Brown, M. and Lowe, D. G. (2007). Automatic panoramic image stitching using invariant features. *International Journal of Computer Vision*, 74(1).

Chen, T., Cheng, M.-M., Tan, P., Shamir, A., and Hu, S.-M. (2009). Sketch2photo: internet image montage. In *SIGGRAPH*.

Chockalingam, P., Pradeep, S. N., and Birchfield, S. (2009). Adaptive fragments-based tracking of non-rigid objects using level sets. In *ICCV*.

Ghanem, B., Zhang, T., and Ahuja, N. (2012). Robust video registration applied to field-sports video analysis. In *ICASSP*.

Granados, M., Kim, K. I., Tompkin, J., Kautz, J., and Theobalt, C. (2012). Background inpainting for videos with dynamic objects and a free-moving camera. In *ECCV*.

Grundmann, M., Kwatra, V., Han, M., and Essa, I. (2010). Efficient hierarchical graph based video segmentation. In *CVPR*.

Joulin, A., Bach, F., and Ponce, J. (2012). Multi-class cosegmentation. In *CVPR*.

Kong, H., Audibert, J.-Y., and Ponce, J. (2010). Detecting abandoned objects with a moving camera. *IEEE Trans. on Image Processing*, 19(8).

Lee, Y. J., Kim, J., and Grauman, K. (2011). Key-segments for video object segmentation. In *ICCV*.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110.

Ma, T. and Latecki, L. (2012). Maximum weight cliques with mutex constraints for video object segmentation. In *CVPR*, pages 670–677.

Ochs, P. and Brox, T. (2011). Object segmentation in video: A hierarchical variational approach for turning point trajectories into dense regions. In *ICCV*.

Rother, C., Kolmogorov, V., and Blake, A. (2004). Grabcut: Interactive foreground extraction using iterated graph cuts. *ACM Trans. On Graphics*, 23.

Sole, J., Huang, Y., and Llach, J. (2007). Mosaic-based figure-ground segmentation along with static segmentation by mean shift. In *SIP*.

Thomas, B. and Jitendra, M. (2010). Object segmentation by long term analysis of point trajectories. In *ECCV*.

Tsai, D., Flagg, M., and Rehg, J. (2010). Motion coherent tracking with multi-label mrf optimization. In *BMVC*.

Yang, L., Guo, Y., Wu, X., and Li, S. (2011). An interactive video segmentation approach based on grabcut algorithm. In *CISP*.

Zhang, D., Javed, O., and Shah, M. (2013). Video object segmentation through spatially accurate and temporally dense extraction of primary object regions. In *CVPR*.