ABSTRACT

Object segmentation in videos or image sequences is a crucial task and has gained more and more attention in the past years, as computer performance is increasing. Segmentation may be carried out in the field of video surveillance with automatic object tracking or as a composing and editing task in the postproduction of movies. In the first case a suitable real-time performance is wanted with a drawback in segmentation accuracy, whereas in the latter one accuracy is the most important factor. We will show a new approach for dynamic segmentation and tracking of objects in image sequences that is based upon graph-based segmentation techniques from still image segmentation. Although the current performance of our algorithm is far from being real-time, we think there is enough room left for optimization in our implementation. We demonstrate the usability of our new MovieCut algorithm on some examples and give some hints to improve the overall performance.

1. INTRODUCTION

Plenty of work has already been done in the scope of image segmentation or more precisely video segmentation. To name a few there are the classical approaches for segmentation in still images \cite{1} and \cite{2} and more recent ideas like in \cite{3} and \cite{4}. Most of these segmentation algorithms rely on changes in contrast or on grouping the same colors only. Yet there seem to be no real one-and-for-all solution to the problem.

Recent advances show graph-based techniques are superior in terms of quality and user interaction compared to other algorithms \cite{5}, \cite{6}, \cite{7}. Their advantage is to combine contrast and color information into an overall energy minimization criterion. We will explain them in the following section in a little more detail, before we show how our new proposal fits into this evolution.

The following section gives a short introduction into the MorphCut algorithm for still image segmentation on which the proposed algorithm for image sequence segmentation is based upon. In the third section follows the core description for the new frame-by-frame segmentation algorithm, called MovieCut. Some experimental results and examples show the usage of our proposed method for segmentation of objects in image sequences in the succeeding section. Finally, we conclude with a summary and give a brief outlook on future work.

2. GRAPH-BASED SEGMENTATION

The idea of using an energy minimization technique for image segmentation and solving it with graph-based algorithms was first described by Greig, Porteous and Seheult in 1989 \cite{8}. We will first describe our MorphCut algorithm \cite{7} that is based upon GraphCut by Boykov and Jolly \cite{5} and GrabCut published in 2004 by Rother, Kolmogorov and Blake \cite{6}.

2.1 Energy Minimization

MorphCut \cite{7} combines the two already known approaches for image segmentation: algorithms based on colors and segmentation based on the contrast in different regions of an image. For successful segmentation the energy formulation

\[
E(z) = P(z) + \gamma \cdot C(z)
\]

has to be minimized. It consists of the two terms: \(P(z)\) for the influence of colors and \(C(z)\) for the impact of different contrast, which will be described in more detail in the following. The weighting parameter \(\gamma\) controls the importance of one term over the other.

2.2 Color Distribution Model

The fidelity term \(P(z)\) gives rise to a cost function, which penalizes false classification of a pixel \(z\) of the image \(I\) to the foreground \(\alpha = 1\) or to the background \(\alpha = 0\). Gaussian mixture models (GMM) are used for approximating the distribution of pixel colors. Background and foreground are each described with five full-covariance Gaussian components \(M_{z,k}\) (see figure 2).

Since the user provides a so-called trimap, where two regions - sure background \(T_B\) and (optional) sure foreground \(T_F\) - has to be defined, one can easily calculate a probability distribution and cost functions \(p_{z,\alpha}\) from the superposition of the Gaussian components

\[
P(z) = \sum_{\alpha} p_{z,\alpha} \cdot
\]

Costs can be calculated from the negative log-likelihood of the probability belonging either to the foreground or to the background. Each Gaussian component is defined as

\[
M_{z,k} = \frac{1}{2\pi\sqrt{\Sigma_k}} \exp \left( -\frac{1}{2} (I_z - \mu_k)^T \Sigma_k^{-1} (I_z - \mu_k) \right)
\]

where \(k\) ranges from 1 to 5 for each component and the term \(I_z\) reflects a three-valued RGB color of the pixel \(z\). The
\( \mu_k \) are the mean color of each component and \( \Sigma_k \) are full-covariance matrices reflecting color dependencies between the three color layers.

Adaptation of the probability distributions \( M_{z,k} \) to the RGB colors is carried out with the iterative expectation maximization (EM) algorithm \([10]\), according to the predefined trimap given by the user. We come back to this point later.

### 2.3 Neighboring Contrast

The prior term \( C(z) \) representing the pair wise interactions between neighboring pixels is calculated from the contrast between each two neighboring pixels \( z \) and \( \hat{z} \):

\[
C(z) = \sum_{(z,\hat{z}) \in N} c_{z,\hat{z}},
\]

where the norm \( ||I_z - I_{\hat{z}}|| \) is the Euclidian distance in RGB space and \( ||z - \hat{z}|| \) indicates the spatial (Euclidean) distance between two neighboring pixels \( z \) and \( \hat{z} \).

Color values of pixel \( z \) are in the range \( 0 \ldots 1 \). The variance \( \sigma^2 \) over all differences in intensity can be seen as the noise floor present in the image. Choosing this parameter carefully lets the contrast term successfully switch between almost zero for high contrast and one vice versa. However, other functions, separating noise from real contrast in the same manner, are also possible.

### 2.4 Minimum Cut / Maximum Flow

From these two properties of each pixel - one belonging to the object or the background, the other being an edge or not - an undirected graph is built \([9]\). More precisely a so called S/T-graph is built, where the two terminals \( S \) and \( T \) represent the object respectively the background (see figure 1). Edges from and to these terminals are weighted with the corresponding foreground/background costs \( p_{z,a} \). Neighboring pixels are connected with edges in 8-way neighborhood, weighted with the corresponding contrast terms \( c_{z,\hat{z}} \).

Using a standard minimum-cut/maximum-flow (MC) algorithm has been proven to give the optimal segmentation border in terms of the energy formulation \( E(z) \) defined in (1). The segmentation border corresponds to the edges representing the minimum cut in the graph.

### 2.5 Iterative Optimization

Due to the iterative nature of the EM algorithm, the whole algorithm is laid out in an iterative way: after each EM iteration, an S/T-graph is built and solved with the minimum-cut algorithm. The resulting segmentation border is used to update the trimap describing foreground and background regions. This new trimap is used for the next EM iteration and so on.

### 2.6 Slow Start

In case of some image materials, the color information is very low, and the segmentation result gets worse. This occurs for example in the case of ship images. In these cases the color information is very low, due to the gray-valued ship body in the blue-gray to blue-greenish water surrounding the ship.

This struggles the EM adaption of the Gaussian Mixture Models in some cases, and so we proposed to use a single EM-step in the beginning without changing the segmentation border - or in other words without changing the trimap. After this pre-iteration of one EM learning step, normal iterations - alternating between EM update and solving for the minimum cut - are carried out.

### 2.7 Contour Morphing

But these ship images gives rise to a second difficulty: due to the surrounding water which is often disturbed by a very turbulent texture and interrupted by waves or spray, the contrast in the image is very high, leading to a fragmented and unpleasant segmentation result. We used morphological operations between succeeding iteration steps to overcome these problems leading to a more stable result in the final segmentation result.

The usage of morphological operations is twofold: at first we use a dilation operation to widen the calculated segmentation border after each iteration step. The basic algorithm only serves in a one-way manner: if a pixel is already classified as
background region, there is no way back for it to become an object pixel again. This is one-way behavior is now slightly intercepted by the widening dilation operation, in which pixels near the segmentation border may be re-classified as object pixels again. We use a simple disc as structuring element.

Of course, this widening has to be adapted to the current iteration level: in the first little iteration, the dilation mask may be much bigger, than in the later iterations, when the segmentation is already near its local optimum. We use an exponential decreasing size of the dilation mask, but other techniques might also be suitable in other applications.

As a nice profit, this dilation operation also serves as slowdown in the convergence of the segmentation algorithm, giving the Gaussian Mixture Models even more time to adapt themselves to the low color information persistent in the images. The convergence is slower - i.e. we need more iterations to lead to a suitable result - but the segmentation result is also more stable and reliable as before.

Second, other morphological operations might (in some cases) also be usable, depending on the image content one wants to process. Using the opening&closing operation is very useful in the case of cluttered and fragmented segmentation result. This is again seen as a stabilization operation to the segmentation algorithm.

Although opening&closing is a combination of dilation and its counterpart erosion, we cannot combine these two steps, because different mask sizes for the structuring element (a disc in both cases) are used: a decreasing size in the first case as described above and a constant size for the latter, depending in the image content (i.e. the segmentation result).

3. MOVIE-SEGMENTATION

To apply the previously described algorithm to movies or images sequences, one takes advantage from the iterative nature of MorphCut. Instead of interpreting each frame as a single image and segment any object by a whole segmentation run, only a single iteration is used on each single frame. This means, consecutive iterations of the segmentation algorithm are processed on consecutive frames of the image sequence. Whereas the basic idea is simple and straightforward, several modifications have to be made to the image segmentation algorithm to handle image sequences. We will describe these modifications step-by-step in the following to highlight the difference between segmentation in still images and in image sequences.

3.1 User Selection of the Region-of-Interest

First of all a suitable region of interest (ROI) has to be selected by the user, where the object lies in. Like in the previously described still image segmentation algorithms, the user has to select at least a background region. But for image sequence segmentation this region is no longer used as the sure background region \( T_B \), but instead the inverse of it - the inside region - is used as the undefined region \( T_U \) in the following.

Since the image content may change over time, there is no longer use of the full trimap used for sure background \( T_B \), sure foreground \( T_F \) and the unknown region \( T_U \). In fact only the unknown region is used in our MovieCut implementation, in the sense that it reflects the current object mask - cutting the object from the background.

But additional information based on user interaction might still be useful in some applications. As already men-

Figure 2: A snapshot of the evolution of the Gaussian Mixture Models in the church sequence in false color: the blue colors show pixels assigned to the five background components, the red pixels are assigned to one of the five foreground components. In the upper right corner a fragmented miss-segmentation can be seen clearly.

3.2 Gaussian Mixture Models

As already described in the MorphCut algorithm, we use a single pre-learning step of the EM adaption of the Gaussian Mixture Models. As stated above, there are no sure background or foreground regions anymore, which originally serve as the basis for the EM adaption to the underlying color model. In this case, the color model has to be (almost) learned in one single step based on the initially given background region by the user.

During the algorithm we can update the Gaussian Mixture Models by the result of each iteration step. After the minimum-cut solution background and object parts are separated for this frame and the color distribution can be updated accordingly.

As long as the color model does not change dramatically in the scene visible in the image sequence, a single EM step for each frame is sufficient to re-learn the small changes. In the case of fast camera movements or other drastic changes, this slow EM learning might fail and has to be updated more frequently (i.e. more EM-steps per iteration or frame).

3.3 Updating the Contrast-Terms

In the case of still images the contrast terms - the \( c_{z \xi} \) in equation [5] - can be calculated once for all iteration steps, since these values do never change. Now we have different image frames in each iteration steps and so we also have to update the prior term before each S/T-graph can be built.

This leads to a slight increase in terms of computational operations needed in each iteration for calculating the derivation in equation [6].
3.4 Contour Morphing

Since the object that shall be segmented out of the image frames might move around in the image scene and even the viewpoint - showing the object from different directions - may change also, the segmentation border must be post-processed after each iteration.

We made again use of the morphological dilation operation as in the MorphCut algorithm. This helps to increase or widen the contour line a few pixels, therefore giving the EM learning and the graph-based minimization a new starting point to work for the next iteration. We again overcome the original one-way assignment of pixels either to object or to the background. After the dilation has been carried out - to repeat: only the unknown region is maintained and updated in our MovieCut algorithm - pixels around the segmentation border might be re-assigned during the next iteration step, allowing the object to move these few pixels from frame to frame.

The size of the dilation mask can be seen as a performance parameter: on the one hand, one may want an accurate segmentation result, that result from a small dilation size, letting the segmentation algorithm converging to local minima. On the other hand a fast reaction to any movement or change of the object would reflect in a much bigger dilation mask, but preventing the segmentation border to remain in stable local optima.

3.5 Image Stabilization

In the presence of shaking images due to the use of a handheld camcorder and/or a long focal distance, an image stabilizing algorithm should be applied to the video before processing any segmentation. This would provide a more stable image and object and therefore the size of the dilation mask that is used for morphing and widening the calculated segmentation borders can be chosen much smaller.

Without stabilizing the single frames the dilation mask has to be chosen much bigger to reflect each single jump and movement of the camera that is visible in the image frame by the movement of the whole scene.

3.6 Stopping Criterion

The original still image segmentation algorithms use an energy based criterion to stop the iteration when a good segmentation border is already calculated (i.e. there was no or only a slight improvement in the segmentation result). In the case of image sequences one can argue, that no stopping criterion is needed any more, since we always need a new iteration on the next movie frame.

But in fact one can use a stopping criterion to detect dramatic changes of the scene or the viewpoint of the camera. Since the segmentation mask is made to adapt itself only slowly to any change in the images, one can detect these scene changes or even the disappearing of the object outside the visible frame by inspecting the change in the calculated energy value, which has been minimized by the minimum-cut/maximum-flow algorithm.

4. EXPERIMENTAL RESULTS AND DISCUSSION

Figure 3 show three example image sequences that have been segmented with the proposed MovieCut-algorithm. Since each sequence covers several seconds to minutes in time, we only show eight selected frames of each sequence to show the evolution of the segmentation border over time.

The first showing helicopter flight around a church. The sequence has been made with a handheld video camera and is therefore floating and jumping around a lot, as one can see in the eight example frames. Nevertheless since we iterated over every frame, the difference between consecutive frames is kept small.
frames is quite small and segmentation border has only to be expanded by 15 pixels in each iteration to adapt itself to the camera/object movement. The eight frames are spaced by six seconds each, spanning a total time range of almost one minute. The last frame shows the effect of zooming out the object with the camera which is covered very well by our algorithm.

Only eight selected frames of the whole sequences with a distance of several seconds are shown here, but the segmentation was carried out on each frame (25 frames per second). In the second line only every 10th frame has been used to calculate the segmentation, improving the overall processing time by a factor of ten. Nevertheless, if the movement in the scene is not too large, using every N-th frame still give reasonable results as can be seen in the second row of figure 3.

In the third row we show an image sequence of a moving merchant ship through the water. Although the initial user selection in the first frame only covered the ship itself and as least as possible of the water and spray, the algorithm adapts itself to the color model of the surrounding white splash water (second frame), leading to a more or less satisfying segmentation result. Anyhow, the segmentation remains stable across the following six frames, even when the object is only partially visible in the images. Calculation is carried out on each frame again.

The last row shows a segmented cat that is seeking around the floor. Eight consecutive frames are shown; therefore there is not much overall change in the scene. As can be seen, the segmentation border remains very constant despite the fact, that it is widened between each two frames before the new iteration step is started. Each time the single segmentation step yields a suitable and stable segmentation border.

To summarize, we have three parameters to tune the algorithm performance on different image/video material:
1. The weighting factor $\gamma$ controlling the influence between contrast prior and color distribution.
2. The size and the decrease factor of the dilation mask, that is used for promoting the undefined region $\tilde{U}$ to the next frame.
3. The step rate, i.e. how many frames are skipped between two consecutive iteration steps.

5. CONCLUSIONS & OUTLOOK

We have shown that previously proposed graph-based segmentation algorithms for still images are also suitable for segmenting objects in movies and image sequences. The basic idea is to use subsequent frames in each iteration, therefore making the segmentation adapt smoothly to slight changes in the video scene. Some small modifications have been described that have to be made to the still image segmentation.

The computational load is quite high, since on each frame the Gaussian Mixture Model has to be learned and a S/T-graph is solved via the minimum-cut/maximum-flow algorithm. At the moment we are able to calculate a single iteration on one frame - with a resolution of $720 \times 576$ - in less than 10 seconds, which is far away from real-time. Nevertheless the code is mainly implemented in MATLAB which has been proven to be not the fastest one can do. Implementing the algorithm directly on a video card will surely boost the performance significantly and is therefore the next thing to do.

Another idea to boost the time performance is to use only each 10th frame (or any other step-size) to solve one segmentation iteration. In this case the difference in the used frames between consecutive iterations is quite large and can therefore lead to a disturbed segmentation. One can stabilize this by applying an image stabilization technique first to steady fast camera movements and to stabilize the objects position in the frames.

REFERENCES