Automated Cinematic Background Blur in Videos

Clayton Lie
The Chinese University of Hong Kong, Shenzhen
124040041@link.cuhk.edu.cn

Kaylinn Venezia Kurniawan
The Chinese University of Hong Kong, Shenzhen
124040033@link.cuhk.edu.cn

Paper ID: 10

Abstract

One key element of cinematic videos is background blur, a feature that most phone cameras lack due to their limited depth of field during recording. To overcome this limitation, video editors typically duplicate the video, manually mask the subject, blur the background, and layer them to simulate depth. However, this process is time-consuming and labor-intensive, which is why automating it would save time and make cinematic effects more accessible to users with budget phones. While deep learning tools like Rembg and Mediapipe can automate this, they rely on pre-trained models, which increase computational cost, complexity, and dependency on large datasets. Hence, we propose a lightweight alternative. First, the first video frame is processed through Graph Cut with PyMaxFlow. Next, the first frame will be put into the Template Bank. During the process, all the frames will go through mask propagation. The next frame will be localized with DTTM and NCC, followed by mask refinement using Dense CRF. Then, the result of the refined mask will be updated into the template bank and blurred the background at the same time. Our framework will be tested using DAVIS 2016 and a personal dataset. Finally, an automated cinematic video with a blurred background is provided.

1 Introduction

In cinematography, the depth-of-field or bokeh effect is crucial in increasing the viewing experience of movies. It enhances focus and adds a sense of depth by blurring the background of foreground objects (or vice versa). This is similar to how the human eye perceives objects in real life, which is why having this effect can make videos seem more natural and cinematic.

Unfortunately, not all phone cameras can afford this effect due to their limited depth of field during recording. To overcome this limitation, video editors typically duplicate the video, manually mask the subject, blur the background, and layer them to simulate depth. Nevertheless, this procedure requires a lot of work and time.

In this paper, we propose an automated background blur framework (shown in Figure 1) that is designed to save time and make cinematic effects more accessible to everyone. Our framework consists of three main stages: graph cut, mask propagation, and background blurring. https://github.com/kaylinnvk/cinematic-blur

1.1 Related Work

Deep learning for VOS Gao et al. [5] provided a comprehensive overview of deep learning-based methods for video object segmentation (VOS), highlighting that the improvement of deep learning technology can show good performance in VOS. However, there are some limitations, such as occlusions, ambiguous backgrounds, and a lack of motion information. Our method avoids reliance on heavy CNN architectures, exchanging graph cut [1] and dynamic time-evolving template matching [2] as alternatives for foreground segmentation.

MediaPipe In their work, Lugaresi et al. [6] introduced MediaPipe, a framework built for processing arbitrary data. By integrating machine learning, the framework becomes significantly more efficient and improves performance in object detection. Unfortunately, through experimentation, we found one limitation: the foreground object can't be identified correctly. Hence, we apply the graph cut [1] to segment the exact target object.

2 The Proposed Algorithm

2.1 Pipeline Overview

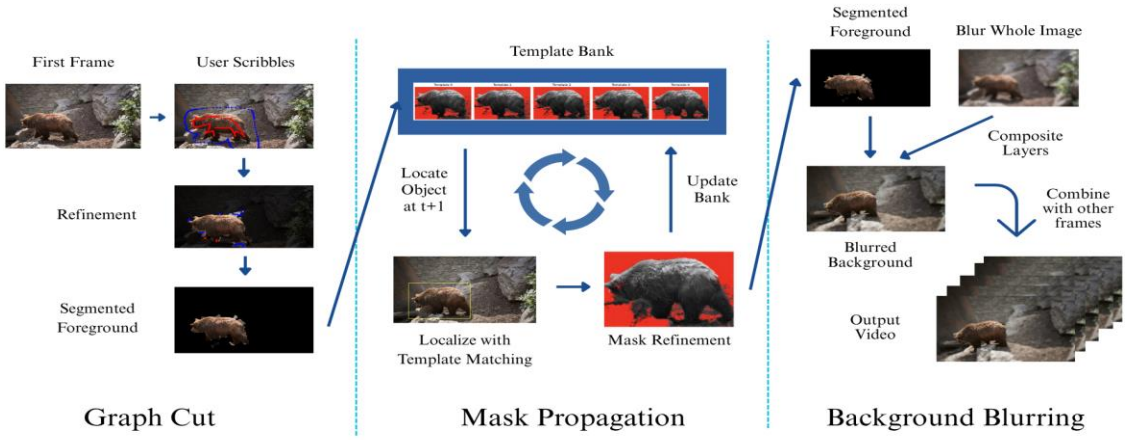

Figure 1: Pipeline Overview of Automated Background Blur in Videos

2.2 Graph Cut Algorithm

The graph cut algorithm [1] is a method used to separate foreground object(s) from the background. The algorithm starts by taking the first frame of the video and segmenting it into superpixels via color-based k-means clustering. Doing this will drastically decrease the number of nodes needed to create the graph, making it computationally lighter. After that, the user will input the scribbles with two different colors: red for identifying foreground and blue for identifying background. Here, each seed will label each corresponding superpixel with the current foreground/background information.

Based on this input, the algorithm will construct a graph, where nodes represent pixels and edges connect neighboring pixels with weights based on their similarity. Additionally, there are two special nodes called the source (representing the foreground) and the sink (representing the background), to which the edges will connect. Lastly, the algorithm computes the "minimum cut" in the graph, which separates foreground and background. To compute the algorithm, we define the energy minimization function:

$$E(f) = \sum_{p \in S} D_p(f_p) + \lambda \sum_{\{p,q\} \in \mathcal{N}} \omega_{pq} \cdot T(f_p \neq f_q) \quad (1)$$

Here, f_p denotes the label (foreground or background) assigned to pixel p, and $D_p(f_p)$ measures how well the labels match data using a Gaussian likelihood function with the initial foreground and background seeds as the inputs. Then, the weight ω_{pq} reflects the similarity between neighboring pixels. Typically, a Gaussian function $\omega_{pq} = e^{-\frac{(I_p - I_q)^2}{2\sigma^2}}$ is used as the weight where I_p and I_q represent color values of sites p and q, respectively, and σ is the standard deviation of the Gaussian. $T(f_p \neq f_q)$ refers to the function that forbids label differences where $T(f_p \neq f_q) = 0$ if $f_p = f_q$ and 1 otherwise. The parameter λ controls the importance of the data term and the smoothness term within the energy function.

As a result, the graph cut will output the segmented foreground. To ensure good results, the segmentation can be refined by the user using scribbles that redefine which superpixel segments are foreground or background. The final segmentation will then be saved into the template bank, a repository for storing templates, as the first template.

2.3 Mask Propagation & Blur Background

We use the dynamic time-evolving template matching (DTTM) mechanism proposed by Huang et al. [2] for localizing the foreground object at the next frame. To improve the mechanism, the fast Fourier transformation normalized cross-correlation (FFT NCC) formula from [3], which also implements NCC using an arbitrary mask to prevent the background from being included in the template matching.

NCC is used to see the similarity between two images. The Fourier domain provides a fast and efficient method for computing correlation using the FFT. Instead of a rectangular template, we use an arbitrary binary mask to ignore background interference. The equations below allow correlation to be computed over non-rectangular regions using binary masks:

$$\mathrm{NCC}(u,v) = \frac{\mathrm{NCC}_{\mathrm{num}}(u,v)}{\sqrt{\mathrm{NCC}_{\mathrm{den},1}(u,v)}\sqrt{\mathrm{NCC}_{\mathrm{den},2}(u,v)}} \tag{2}$$

$$\mathrm{NCC}_{\mathrm{num}} = \mathcal{F}^{-1}(F_1 \cdot F_2^*) - \frac{\mathcal{F}^{-1}(F_1 \cdot M_2^*)\cdot \mathcal{F}^{-1}(M_1 \cdot F_2^*)}{\mathcal{F}^{-1}(M_1 \cdot M_2^*)} \tag{3}$$

Equation (2) refers to the general definition of NCC. The numerator NCC ($\mathrm{NCC}_{\mathrm{num}}$) in Equation (3) computes cross-correlation between inputs F_1 and F_2^* using Inverse FFT, while subtracting the bias caused by masked mean intensities.

$$\mathrm{NCC}_{\mathrm{den},1} = \mathcal{F}^{-1}(\mathcal{F}(f_1 \cdot f_1)\cdot M_2^*) - \left(\frac{\mathcal{F}^{-1}(F_1 \cdot M_2^*)}{\mathcal{F}^{-1}(M_1 \cdot M_2^*)}\right)^2 \tag{4}$$

$$\mathrm{NCC}_{\mathrm{den},2} = \mathcal{F}^{-1}\big(M_1 \cdot \mathcal{F}(f_2' \cdot f_2')\big) - \left(\frac{\mathcal{F}^{-1}(M_1 \cdot F_2^*)}{\mathcal{F}^{-1}(M_1 \cdot M_2^*)}\right)^2 \tag{5}$$

The denominator NCC ($\mathrm{NCC}_{\mathrm{den},1}$ and $\mathrm{NCC}_{\mathrm{den},2}$) in Equation (4) and (5) represents the value to account for image brightness and contrast. $\mathrm{NCC}_{\mathrm{den},1}$ computes the variance of the template patch and $\mathrm{NCC}_{\mathrm{den},2}$ computes the variance of the target region in the current frame. In both cases, the variance is computed under the influence of masking and corrected by subtracting the squared mean.

At each timestep, the current frame is compared against all templates in the template bank. The template with the highest NCC score will be used to calculate the initial mask using similarity and dissimilarity functions. Then, we also use dense conditional random fields (CRF) [4] to further refine the mask of the current frame. The CRF itself is a probabilistic model that models the probability of a set of labels $x = \{x_i\}$ given the observed data I (in this case, the observed data is from the initial mask created) where

$$P(X \mid I) = \frac{1}{Z(I)}\exp\big(-E(x)\big), \tag{6}$$

with Z as the normalization constant. The goal here is to maximize the probability that the labelling is correct by minimizing the Gibbs energy function, similar to the graph cut. The energy function is defined as follows:

$$E(x) = \sum_i \varphi_u\,(x_i) + \sum_{i<j} \varphi_p\,(x_i, x_j), \tag{7}$$

Here, φ_u is the unary potential that represents the negative log probabilities $\varphi_u = -\log P\,(x_i)$, where $P(x_i)$ is the initial probability the pixel i is assigned to label x_i based on the pre-defined foreground/background labelling done by the initial refinement (through similarity and dissimilarity). φ_p is the pairwise potential that models the labelling of pixels i based on j through spatial and appearance consistency. It is modeled as

$$\varphi_p\,(x_i, x_j) = \mu(x_i, x_j)\sum_{m=1}^{K} w^{(m)}\,k^{(m)}(f_i, f_j) \tag{8}$$

with

$$k^{(m)}(f_i, f_j) = w^{(1)} \exp\left(-\frac{|p_i - p_j|^2}{2\theta_\alpha^2} - \frac{|I_i - I_j|^2}{2\theta_\beta^2}\right) + w^{(2)} \exp\left(-\frac{|p_i - p_j|^2}{2\theta_\gamma^2}\right). \quad (9)$$

In the expression, f_i and f_j are feature vectors for pixels *i* and *j*, $w^{(m)}$ represents the weights, and μ is a label compatibility function. The first term in the expression represents the appearance kernel, while the second term represents the smoothness kernel of the Gaussian kernel *k*.

After the mask is refined, it will be updated into the template bank and composited onto the current frame using a Gaussian blur. The parameter values of background blurring (i.e., kernel size, standard deviation, and feather radius) are customizable according to the users' preferences. The mask propagation cycle then continues from DTTM to mask refinement for all the frames. The blurred outputs will then be combined at each iteration into one video as the output.

3 Implementation Details

In this experiment, several fixed parameter values and libraries are used to get the results.

For the graph cut phase, superpixel segmentation is implemented using Skimage's SLIC function. It utilizes color-based k-means clustering to group similar pixels together. Other than that, the graph instance and the maxflow method from PyMaxFlow are used to calculate the min-cut/max-flow of the graph using the Boykov-Kolmogorov algorithm [8]. The number of segmentations for superpixel segmentation is set at 1300 segments; it can be increased to get finer segments, but computationally heavier, or decreased to get coarser segments, but more lightweight.

For the mask propagation phase, the matchTemplate() function from OpenCV is used with the normalized cross correlation set as the method and an arbitrary mask set as the mask. This pre-built function is used instead of implementing a custom one because it is already optimized for template matching. The template bank size is 20 for this experiment, and a thread pool is used to concurrently match multiple templates in parallel threads to speed up the process. Lastly, the Dense CRF is implemented using the PyDenseCRF library in [4].

4 Experiments

To evaluate the effectiveness of our proposed framework, experiments are conducted. We use the DAVIS 2016 dataset and the personal dataset for our experiments.

4.1 Evaluation Dataset

The DAVIS 2016 dataset [7] (Densely Annotated Video Segmentation) is a benchmark dataset designed for evaluating video object segmentation algorithms. It contains 50 high-quality video sequences at a 480p resolution with pixel-level ground truth annotations for foreground objects in every frame. In our work, we utilize the TrainVal (Training and Validation) split, which comprises individual frame sequences and annotation masks. In addition, we introduce a custom dataset that serves as the main core for cinematic videos such as movie scenes. We combine the frame sequences into full videos and use the given first frame of each video for graph cut.

4.2 Evaluation Metrics

The quality of the foreground segmentation can be evaluated through contours and regions using the DAVIS benchmark in [7].

Contour accuracy measures the contour precision between the actual object and the mask. The contour accuracy is defined as $\mathcal{F}$. The average contour alignment ($\mathcal{F}$ Mean) shows the effectiveness of the predicted contour matches the ground truth. The contour recall ($\mathcal{F}$ Recall) measures the frequency of well-aligned

contours across the frames. The temporal drift in contour ($\mathcal{F}$ Decay) indicates the worsening changes in contour alignment.

Region similarity measures the similarity of object segmentation. The region similarity is defined as $\mathcal{J}$ (Jaccard Index) for the *intersection-over-union* (*IoU*) of the segmentation estimation and ground truth mask. The average region overlap ($\mathcal{J}$ Mean) measures the average *IoU* between predicted and ground truth masks. The region recall ($\mathcal{J}$ Recall) captures the frames that exceed the *IoU* threshold. The temporal drift in the region ($\mathcal{J}$ Decay) shows that performance deteriorates over time.

4.3 Evaluation Results

In this section, we evaluate the results of 50 videos from DAVIS 2016 and 22 videos from our dataset. In the dataset, we present $\mathcal{F}$ as contour accuracy and $\mathcal{J}$ as the overall results of region similarity. Each metric measures three evaluations: mean, recall, and decay.

The evaluation for the DAVIS dataset is done on several categories of videos. It is divided into 4 primary categories: trainval, considerable, dynamic movement, fast movement, and occlusion. The trainval category includes the whole DAVIS 2016 dataset. The considerable category consists of scenes that do not have dynamic movement, fast movement, or heavy occlusion, making them easier to segment. The dynamic movement categories include foreground objects whose movement is dynamic or non-rigid movements, like dancing and parkour for example. The fast movement category includes videos that involve rapid and fast movement. Lastly, the occlusion category consists of scenes where the foreground object is partially or fully obscured by other objects.

Figure 2: Results of VOS on different categories of the DAVIS dataset. Every row consists of the highlighted segmented foreground of every 10 frames (from the first 50 frames) of the video. Each row consists of video frames from each category: first row - considerable, second row - dynamic, third row - fast, and fourth row - occlusion.

Video Sets (DAVIS 2016)	Jaccard Index ($\mathcal{J}$)			Contour Accuracy ($\mathcal{F}$)		
	Mean	Recall	Decay	Mean	Recall	Decay
trainval	0.328	0.362	0.337	0.285	0.283	0.313
considerable	**0.569**	**0.692**	0.315	**0.471**	**0.477**	0.278
dynamic	**0.074**	**0.069**	0.283	**0.093**	**0.083**	0.348

fast	0.198	0.176	**0.371**	0.174	0.121	**0.359**
occlusion	0.273	0.271	**0.259**	0.255	0.294	**0.219**

Table 1: Results of VOS on different categories of the DAVIS dataset (Red: Worst performance, Blue: Best performance)

Table 1 shows the evaluation results of the foreground segmentation on the DAVIS 2016 dataset. The mean results on the trainval set (consisting of all 50 images from the dataset) indicate moderate segmentation accuracy with noticeable decay. Most of the highest scores are obtained in the considerable category, suggesting that the algorithm can handle segmenting objects with simple movements and similar appearances throughout the video effectively. In contrast, the dynamic (motion) category has the worst degradation with lowest recall and high decay, showing temporal instability. Similarly, the algorithm cannot handle videos with fast (motion) category, as the results of contour accuracy and region similarity are poor. The occlusion category gives a moderate result yet the lowest decay, suggesting the algorithm can handle objects with partial visibility loss.

The evaluation results on Table 1 show that the algorithm struggles to identify the target object in certain categories, such as those with fast movement, dynamic movement, and occlusion. These categories can lead to spatial misalignment and generate a poor mask, which causes wrong or no detection in the next frames. The imperfect mask refinement also plays a huge role, as it affects the reliability of template matching, giving inaccurate foreground detection. The low scores in the dynamic category suggest that the algorithm is not robust to non-rigid motion. Despite the moderate scores in the occlusion category, the algorithm can maintain temporal stability even when there is visibility loss.

Due to the lack of ground truth segmentations for our personal datasets, only qualitative analysis is performed in this report.

Figure 3: Results of VOS on the personal dataset. Every row consists of the highlighted segmented foreground of every 10 frames (from the first 50 frames) of the video. The first two rows are personal videos recorded by a phone camera. The last two rows are scenes from the *Harry Potter and the Sorcerer's Stone* and *Spider-Man 1* movies.

The qualitative results in Figure 3 demonstrate the algorithm's varying performance across different video scenarios. It is clearly shown in the first two rows that the Region of Interest (ROI) is getting smaller as the size of appearances changes. Moreover, the second row shows that the mask fails to follow the object as the person swings to the camera due to an occlusion. In the third and fourth row, the masks aligned with the shapes of the objects for most frames. Overall, the results show that the algorithm performs well in stable and noticeable shapes (considering the considerable category), but struggles with fast and dynamic movement, as well as occlusion.

5 Discussions

5.1 Limitations

One major limitation of the proposed method is that the dense CRF fails to adapt to changes in the shape or position of the foreground object, such as dynamic and fast movements, rotation, or the appearance of new parts. As a result, the foreground mask may become overly suppressed, leading to a shrinking ROI and failed detections in future frames. Moreover, using a dense CRF is also computationally heavy. The algorithm still lacks adaptability in segmenting foregrounds with dynamic and fast movements, as well as thick occlusions. Template matching struggles to track non-rigid objects, causing the templates and search patch to be cropped occasionally.

5.2 Implication & Future Direction

Our works indicate that there is still a need for improvement for contour accuracy and region similarity. Future work still needs to explore adaptive alternatives for Dense CRF and investigate a better method for mask refinement, which enhances robustness. Integrating deep learning can indeed increase the accuracy of segmentation.

References

[1] B. Peng, L. Zhang, and J. Yang, “Iterated graph cuts for image segmentation,” in *Lecture notes in computer science*, 2010, pp. 677–686. doi: 10.1007/978-3-642-12304-7_64.

[2] X. Huang, J. Xu, Y.-W. Tai, and C.-K. Tang, “Fast video object segmentation with temporal aggregation network and dynamic template matching,” *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8876–8886, Jun. 2020, doi: 10.1109/cvpr42600.2020.00890.

[3] D. Padfield, “Masked Object Registration in the Fourier Domain,” *IEEE Transactions on Image Processing*, vol. 21, no. 5, pp. 2706–2718, May 2012, doi: https://doi.org/10.1109/tip.2011.2181402.

[4] P. Krähenbühl and V. Koltun, “Efficient Inference in Fully Connected CRFs with Gaussian Edge Potentials,” in *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, vol. 24, 2011, pp. 109–117.

[5] M. Gao, F. Zheng, J. J. Q. Yu, C. Shan, G. Ding, and J. Han, “Deep learning for video object segmentation: a review,” *Artificial Intelligence Review*, vol. 55, pp. 8973–9027, 2022, doi: 10.1007/s10462-022-10176-7.

[6] L. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Hays, F. Ricci, M. Wong, J. Klee, M. Steiner, and M. Grundmann, “MediaPipe: A Framework for Building Perception Pipelines,” *arXiv preprint arXiv:1906.08172*, 2019.

[7] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. Van Gool, M. Gross, and A. Sorkine-Hornung, “A benchmark dataset and evaluation methodology for video object segmentation,” in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, 2016, pp. 724–732.

[8] Y. Boykov and V. Kolmogorov, "An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 26, no. 9, pp. 1124–1137, Sept. 2004, doi: 10.1109/TPAMI.2004.60.