Automatic Logo Removal for Semitransparent and Animated Logos

Erofeev Mikhail, Dmitriy Vatolin
Department of Computational Mathematics and Cybernetics
Moscow State University, Moscow, Russia
{merofeev, dmitriy}@graphics.cs.msu.ru

Abstract

Adding a visual logo to a video sequence is a popular method of identifying the owner of that sequence. In this paper we propose a fully automatic method for removing opaque, semitransparent and animated logotypes from video sequences.

Keywords: *Logotype removal, video processing.*

1. INTRODUCTION

In this paper we consider logo removal as a process of automatic detection of a logotype's shape and position in each frame of the video, followed by complete removal of that logotype from the video. The most common types of logos are the following:

1. Opaque static logo—the logo image is overlaid on all video frames
2. Semitransparent static logo—the source frame is alpha blended with the logo image
3. Animated logo—the logo image changes with time, but usually this change is periodic

The proposed method allows for removal of all these types of logos.

2. RELATED WORK

Several related works address TV logos. These works can be classified into two categories.

2.1 Logo detection using a logo database

Methods in this group use previously collected information about a set of logos to detect which one is shown on the screen.

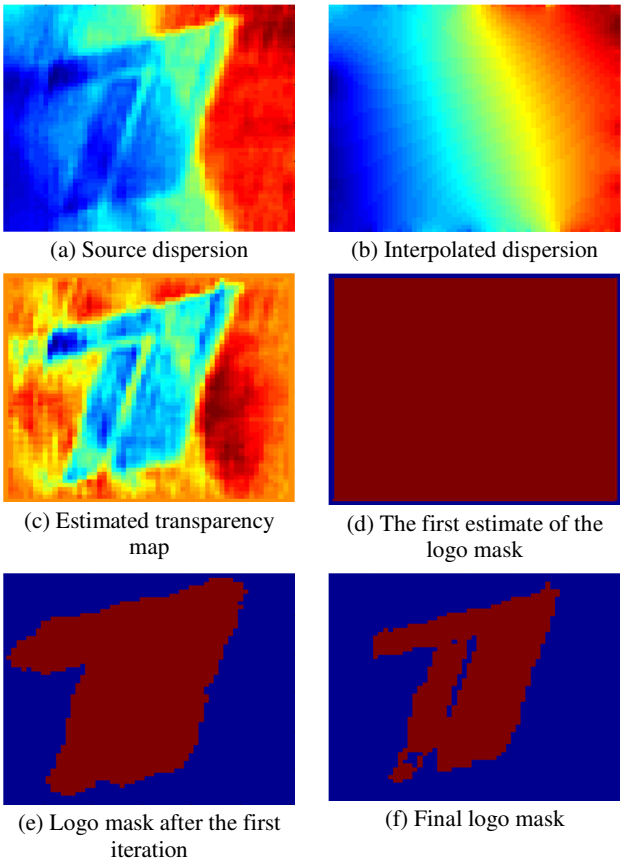
In [3], the authors propose a real-time approach to detecting opaque, semitransparent and partially animated logos. They also propose using average gradient values to detect semitransparent logos. Our approach uses average gradient values to estimate the position of a static logo. In [1], the authors reported good detection results for animated logos when using the unified logo boundary representation.

Nevertheless these approaches cannot be applied in the case of arbitrary logotypes, because of the need to build a database containing all existing logos.

2.2 Logo detection and removal with assumptions

The second group of works solves the problem of automatically removing an arbitrary logotype from a video sequence. The authors in these cases make some additional assumptions about the logo (for example, the logotype doesn't change its color or shape) to permit detection of the logo in the video data.

In [2], the authors propose methods for detecting and removing semitransparent and opaque logos. They assumed that the video content—except for the logo—changes over time. To exploit these temporal variations for logo detection, the sequence is divided into segments with static content. A similar concept is used in our approach to acquire a high-contrast dispersion map.



(a) Source dispersion (b) Interpolated dispersion

(c) Estimated transparency map (d) The first estimate of the logo mask

(e) Logo mask after the first iteration (f) Final logo mask

Figure 1: Main steps of building logo mask

In [4], the authors use a multi-stage approach to logo detection. First, simple binary segmentation is performed, thereby filtering out a large portion of the pixels that belong to moving objects and to the background. The second step involves a Bayesian classifier and neural networks to detect the "coarse" logo. The third step consists of some post-processing to refine the logo mask.

Both of these approaches fail to remove animated logos.

3. PROPOSED APPROACH

Our method consists of two main steps: logo detection and logo removal. For the first step we determine the exact position of the logo and its binary mask. Also for semitransparent logos, some additional information about their color and transparency is collected. The second step uses the data from the previous step to completely remove the logo from the image

3.1 Logo detection

Our method treats static and animated logos in fundamentally different ways. Static opaque and static semitransparent logos are handled in the same way. We discuss the cases of static logos and animated logos separately.

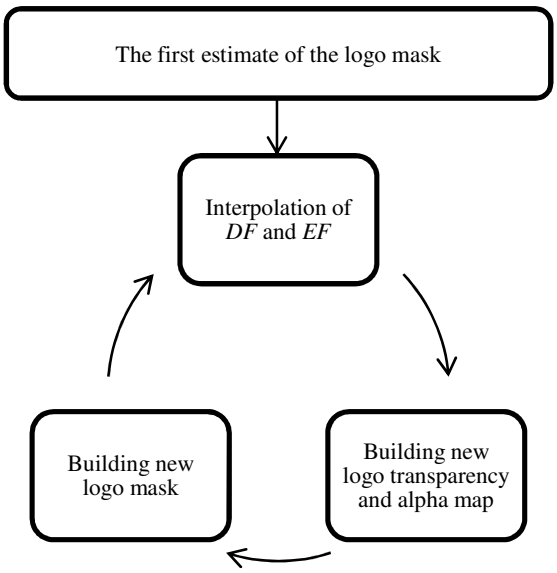


Figure 2: Algorithm of building logo mask

3.1.1 Static logo

For static logos, the frame containing the logo can be considered a source frame that is alpha blended with logo image.

$$I(t) = (1 - \alpha)F(t) + \alpha L$$

Here *I(t)* is the frame containing the logo, *F(t)* is the source frame and α is the transparency map. For opaque logos, the transparency map consists only of the value 1 for points that are part of the logo and the value 0 for all other points.

The first step in logo detection is estimating the logo's position. For this purpose we calculate a gradient field for each frame and then compute the average gradient map for several frames.

$$G(n) = \frac{1}{n}\sum_{i=1}^{n} \|\nabla \mathrm{I}(ki)\|$$

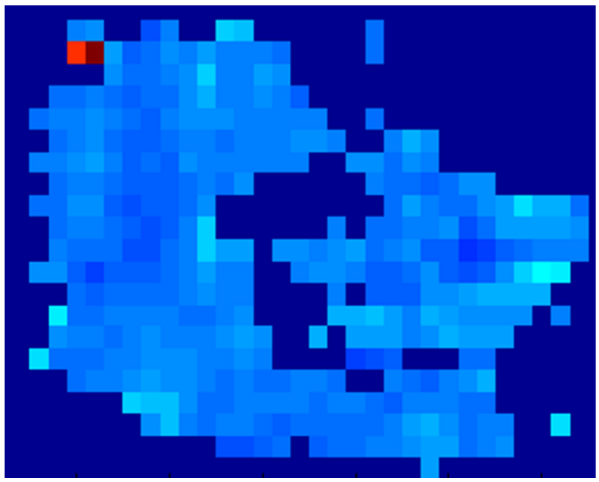
Here k is a time scaling factor (k=20 in our experiments); such time scaling allows us to dismiss the edges of slowly moving objects. For each *G(i)* we determine a set bounding boxes containing areas with high values (the value should be higher than that of 80% of the other *G(i)* points). If a bounding box doesn't change size and position for several instances of *G(i),* that bounding box is assumed to contain a logo image.

All computations at this point are carried out inside the selected bounding box. The last step of logo detection is intended to build the final binary mask of the logotype, its transparency map and its color. We collect a set R of n (n=200) logo region images that are as different as possible. For each video frame we cut out a logo region r. If R contains less than n images, r is inserted into the set. If R already has n images and the following condition is true

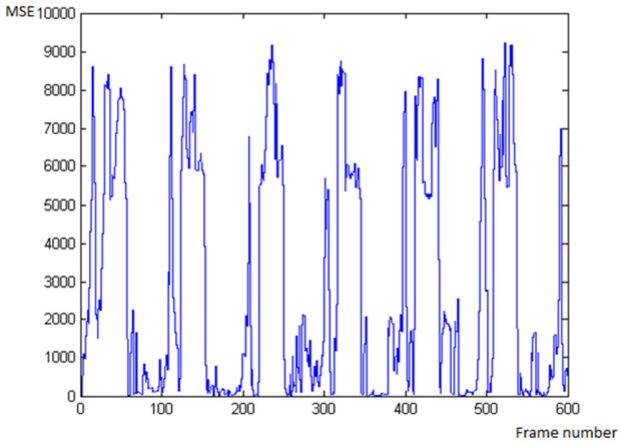
$$\min_{r' \in R} \|r - r'\| > \min_{r_1 \in R, r_2 \in R. r_1 \neq r_2} \|r_1 - r_2\|,$$

then r is inserted into R and the set of the most similar regions R_s is constructed. Then we randomly select one region from the set R_s and remove it from R.

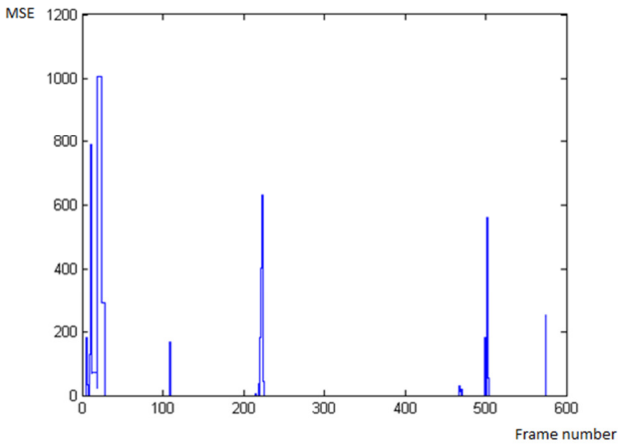
$$R_s = \arg\min_{r_1}\left(\min_{r_2 \in R, r_1 \neq r_2} \|r_1 - r_2\|\right)$$



(a) Period-quality measure for each block
(blocks with higher quality have brighter color)



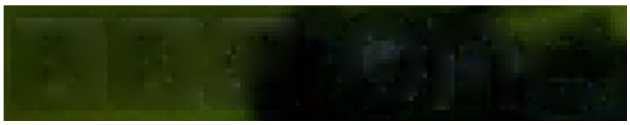
(b) Distances function for block containing an animated logo



(c) Distances function for block containing no logo

Figure 3: Example of quality and distance functions

(a) Source logo region



(b) Spatial interpolation

(c) Logo subtraction

(d) Motion estimation

Figure 4: Source region and examples of several methods of reconstructing frames

For all images in *R,* dispersion and mathematical expectation in the time domain are computed (an example of dispersion is shown in fig. 1(a)).

Now we make assumption that dispersion and mathematical expectation of the whole video are equal to dispersion and mathematical expectation of *R*.

$$DI = DR$$
$$EI = ER$$

Next, the iterative process of building the logo mask is carried out. Fig. 2 shows the flowchart for this algorithm. The first estimate of the logo mask is the rectangle covering the entire logo region, except for a single-pixel-wide border around it, as shown in fig. 1(d). Taking into account only points not covered by the logo mask, we interpolate other points in the dispersion map by solving Laplace's equation.

$$\Delta DF = 0$$

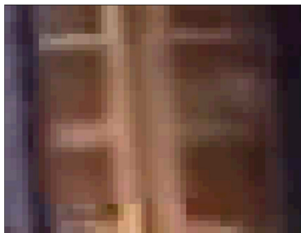
The solution *DF* of this equation is the interpolated dispersion (an example of which is shown in fig. 1(b)) behind the logotype. *EF* is interpolated in the same way.

We can compute a transparency map (an example of which is shown in fig. 1(c)) and color for each logo point.

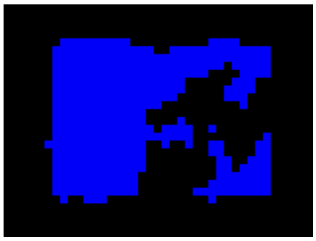
$$DI(t) = D[(1-\alpha)F(t) + \alpha L]$$
$$\mathrm{EI}(t) = \mathrm{E}[(1-\alpha)F(t) + \alpha L]$$
$$\alpha = 1 - \sqrt{\frac{DI(t)}{DF(t)}}$$
$$L = \frac{EI(t) - (1-\alpha)EF(t)}{\alpha}$$

Using simple binary segmentation with a threshold of $\alpha > 0.2$, we can more accurately estimate the logo mask (an example of which is shown in fig. 1(e)). The new mask is used to compute a new transparency and color map. In most cases three iterations of this process are enough to obtain an accurate logo mask (an example of a final mask is shown in fig. 1(f)).

(a) Source logo region



(b) Reconstructed



(c) Estimated logo mask

(d) Estimated logo mask

Figure 5: Source and reconstructed regions from video sequence with animated logo. (c-d) masks for different phases of animated logo

3.1.2 Animated logo

Most animated logos change their color, shape and position periodically. The first step of our approach for animated logo detection is estimation of this period. To this end we divide the frame into regular blocks $B_{i,j}^t$ (where i and j are spatial coordinates and t is the frame number) that are 16×16 pixels each. For each block we estimate its period of change $T_{i,j}$ and period-quality measure $Q_{i,j}$.

$$T_{i,j} = \arg\min_t \sum_{t'} \left\| B_{i,j}^t - B_{i,j}^{t'} \right\|$$
$$Q_{i,j} = \left(\min_t \sum_{t'} \left\| B_{i,j}^t - B_{i,j}^{t'} \right\| \right)^{-1}$$

To save time our method doesn't compute these values directly. We compute the function of distances between blocks for the first frame and a subsequent frame.

$$d_{i,j}(t) = \left\| B_{i,j}^1 - B_{i,j}^t \right\|$$

We can assume that

$$\left\| B_{i,j}^{t_1} - B_{i,j}^{t_2} \right\| \approx \left| d_{i,j}(t_1) - d_{i,j}(t_2) \right|.$$

Using this assumption, $T_{i,j}$ and $Q_{i,j}$ can be computed much faster.

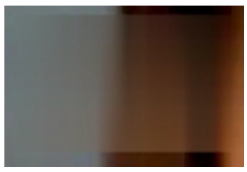
Fig. 3(a) shows an example of $Q_{i,j}$ for each block. Fig. 3(b) illustrates a distance function for the block containing an animated logotype. This function changes periodically, and the block has a large period-quality measure. Fig. 3(c) shows another distance function computed for a block containing no logo. This function changes chaotically, and its corresponding block has a low period-quality measure. The block $B_{i,j}$ with the highest $Q_{i,j}$ is assumed to contain an animated logo with a period of change $T_{i,j}$.

When the period of change is known, we consider frame sets $A_i(t)$.

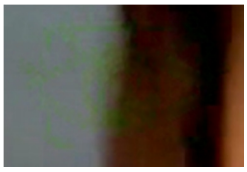
$$A_i(t) = \{ I(Tt + i) | t \in N \}$$

(a) Source logo region

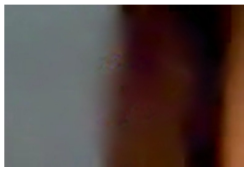
(b) Fuzzy Logo Filter



(c) Logoaway



(d) DeLogo



(e) Our approach

Figure 6: Comparison of logo removal tools

Here T is the estimated period of change. Each A_i can be regarded as a video sequence containing a static logo, and the methods discussed in the previous section can be used for its detection.

3.2 Logo removal

Logo removal is the last step of the proposed approach. We implemented and tested several methods of reconstructing frames behind the logo.

3.2.1 Spatial interpolation

This method uses only the binary logo mask. We solve Laplace's equation to interpolate points covered by the logo mask. This method yields good results when applied to smooth areas, but it fails to when applied to textured areas. An example of the results is shown in fig. 4(b).

3.2.2 Logo subtraction

This method can deal only with semitransparent logotypes and uses the transparency map and logo color data collected in the previous step. To compute the frame without the logo, the following formula is used.

$$F(t) = \frac{I(t) - \alpha L}{1 - \alpha}$$

Fig. 4(c) shows an image obtained using this method.

3.2.3 Motion estimation

This method uses the logo's binary mask and motion information from previous frames. The method can deal with any type of logo. Fig. 4(d) shows results.

4. RESULTS

We tested our methods on several video sequences containing different types of logos. The average frame rate for the logo detection step for HD video was 3, and the average frame rate for the removal step was 4. Fig. 4-5 show example frames for which our approach removed the logo.

Objective comparison of our approach with those of several publicly available logo removal tools was performed. A video sequence without any logotypes was taken as the ground-truth sequence. Opaque and semitransparent logos were added to this sequence. Each tool was used to remove the logo from the test sequences, and the PSNR was measured relative to the ground-truth sequence and its output. Results for this comparison are shown in the table below. Fig. 6 shows output of these tools. Unfortunately we cannot provide any comparison for animated logos because of the lack of a publicly available tool for automatically removing such logotypes.

Filter name	Required additional user input	Opaque logo PSNR (dB)	Semitransparent logo PSNR (dB)
Fuzzy Logo Filter [5]	No	36.29	36.30
Logoaway [6]	Logo bounding box	36.35	36.36
DeLogo [7]	Logo mask	38.64	35.80
Our approach	No	39.38	39.44

As we can see in the table above our approach has the best PSNR value among publicly available tools and requires only video with logotype as input. Fig. 6 also shows that output of our method contains less visually noticeable distortions then output of other methods.

In this paper we have presented logo detection and removal techniques for several types of logo. For static logos we analyze dispersion map of the logo region to carry out logo mask. For animated logos we analyze periodical changes in the input video sequence to determine logo region. Also our animated logo detection method is the first animated logo detection method that doesn't require additional source data.

5. ACKNOWLEDGEMENTS

This research was partially supported by grant 10-01-00697-a from the Russian Foundation for Basic Research.

6. REFERENCES

[1] E. Esen, M. Soysal,T. Ateş, A. Saracoğlu and A. Alatan. *"A Fast Method for Animated TV Logo Detection"*. Content-Based Multimedia Indexing, 2008. CBMI 2008.

[2] K. Meisinger, T. Troeger,M. Zeller, and A. Kaup, *"Automatic TV logo removal using statistical based logo detection and frequency selective inpainting,"* presented at the Eur. Signal Processing Conf., Sep. 2005.

[3] A. dos Santos and H. Kim. *"Real-Time Opaque and Semi-Transparent TV Logos Detection"*. In WACV, 2007.

[4] W. Q. Yan, J. Wang, and M. S. Kankanhalli, *"Automatic Video Logo Detection and Removal"* Multimedia Systems, 10(5), pp. 379-391, July 2005.

[5] http://wiki.atrox.at/index.php/TV_LogoRemove

[6] http://www.videohelp.com/tools/Virtualdub_Logoaway_filter

[7] http://neuron2.net/delogo132/delogo.html