Real-Time Opaque and Semi-Transparent TV Logos Detection

Alex Reis dos Santos, Hae Yong Kim
Escola Politécnica, Universidade de São Paulo, Av. Prof. Luciano Gualberto, trav. 3, 158,
CEP 05508-900, São Paulo, SP, Brazil
alex.santos@ibope.com.br, hae@lps.usp.br

Abstract [1] **- One of the most important strategies created by TV broadcast stations to claim and protect video content ownership is the TV logo. With different colors, shapes and styles, TV logos identify the broadcast station, and sometimes even the kind of the broadcasted program. Consequently, the detection of TV logos is useful in applications ranging from detection of TV commercials to audience measure. In this paper, we propose an improved edge-based template matching to detect opaque, semi-transparent and partially animated logos. The proposed algorithm could be implemented in a DSP for real-time logo recognition, because it is economic in memory use and requires low processing power. We tested during more than 24 hours our implementation to recognize logos of Brazilian and American TV channels and achieved 100% of correct recognition rate.**

I. INTRODUCTION

One of the most important strategies created by TV broadcast stations to claim and protect video content ownership is the TV logo. These logos can be considered as a visible watermark and identify the broadcast station, and sometimes even the kind of the broadcasted program. For example, some channels change their logo from semi-transparent to opaque to indicate alive transmission. Also in most channels, the logo disappears during transmission of TV commercials.

Consequently, a good way for audience surveillance institutes to detect the channel selected by a TV viewer is by finding TV logos out of the video stream [1]. Another application, proposed in [2, 3], is to use the lack of logos to indicate the presence of a television commercial. The logo detection is the first step to remove it, using inpainting techniques [4, 5, 6], where the viewing experience can be improved with the logo removed.

TV logos can be classified in three types: opaque, semi-transparent and (partially) animated. Figure 1 depicts some examples. On some Brazilian broadcast stations, the logo changes from semi-transparent to opaque when the content is being transmitted alive, for example in Record channel (second row of figure 1).

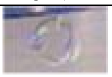
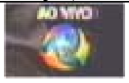
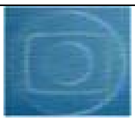
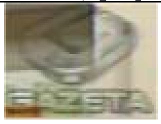
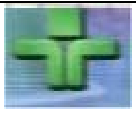
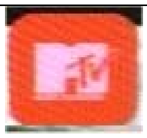
Discovery (animated)	Discovery (animated)
 Record (semi-transparent)	 Record (opaque)
 Globo	 Gazeta
 Cultura	 MTV

Fig. 1: Examples of captured TV logos: Discovery logo is partially animated (the terrestrial globe spins).

The aim of this paper is to measure TV audience by finding a logo in the video stream. We search the video

[1] The authors would like to express their gratitude to IBOPE and CNPq for the partial financial supports of this work under grants 305065/2003-3 and 475155/2004-1.

stream looking for one (or more) logo out of a set of previously stored TV logos. We propose a single algorithm to detect opaque, semi-transparent and partially animated logos. The proposed algorithm is economic in the memory usage, and requires low processing power, so that it could be implement in a portable DSP board. Opaque logos are easy to detect. Partially animated logos can be regarded as opaque ones, because they can be detected through their immovable parts.

Semi-transparent logos are the most difficult ones to detect. Our method first discards the color information, because the hue and the color saturation information do not contribute in the task of detecting semi-transparent logos, due the changing background. Then, we proceed with time averaging (as mentioned by Albiol *et al.* in [2]). The time averaging emphasizes the logos while blurs out the background. Next, we discard the absolute grayscale by extracting the edges, because the grayscale of semi-transparent logos vary with the background, while the edges do not.

Even using only the grayscale edge information, we achieved 100% of detection accuracy. Our result is in apparent contradiction with Wang et al.'s paper [7], where they state "edge-based template matching is weak for semi-transparent ones when incomplete edges appear."

II. Extracting edges

A video stream in PAL-M system (Brazilian broadcast standard) has 493×720 pixels. Empirically, we observed that the logos always appear inside 4 rectangular regions located near the four corners. So, we extracted four sub-videos (say, F_1, F_2, F_3 and F_4) each one with 100×150 pixels. Figure 2 depicts the four sub-regions and figure 3 depicts a frame of the video stream F obtained by concatenating the four sub-videos. For other TV systems with different pixel resolutions (such as NTSC), a slightly different set of sub-regions must be defined.

Empirically, we have observed that each TV logo can appear in only one or two sub-videos. For example, CNN logo always appears in the upper left sub-video F_1, and SporTV logo always appears in the upper right sub-video F_2 or in the lower right sub-video F_4. Consequently, it is necessary to search for CNN logo only in the sub-video F_1, and for SporTV logo only in the sub-videos F_2 and F_4. We have built a table containing the information of in which sub-videos each logo can appear. This process accelerates the processing, essential for real-time implementation.

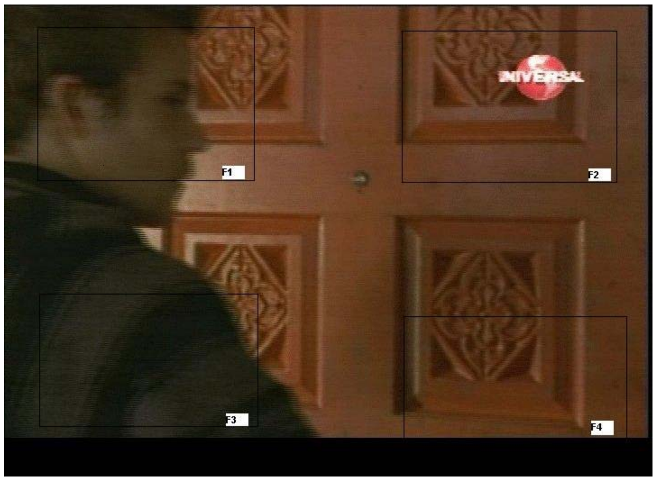


Fig. 2: Extraction of the four sub-regions where TV logos can appear.

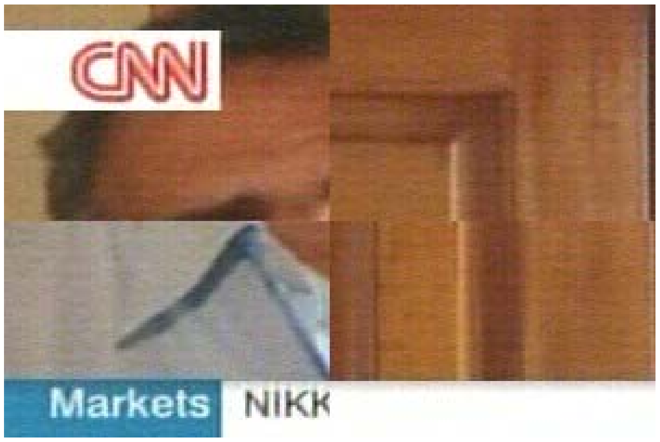


Fig. 3: The four sub-regions merged to form video stream F.

We discarded the color information, because the color of a semi-transparent logo changes with the background variation.

Time averaging is used to emphasize the pixels that either do not vary through the time or vary only a little. This filtering removes the inconstant background images and emphasizes the logos. Only one out of Δt frames are taken into account in the time averaging. Δt is approximately 30 frames (1 second), and the processor uses this time interval to make the rest of the processing (edge extraction, logo searching, etc.) Indeed, we have noticed that it is not worth to use all frames in the time averaging, because averaging similar frames does not help to get rid of the background. Mathematically:

$$\bar{F}(j,k,t) = \frac{1}{2}\left[\bar{F}(j,k,t-\Delta t) + F(j,k,t)\right],\ t \geq 0 \quad (1)$$

where $\bar{F}(j,k,t)$ is the time-averaged video stream at pixel (j, k) and frame t. Note that the time-averaged frame at time t is a weighted average of the frame t (with weight 0.5), the frame $t-\Delta t$ (with weight 0.25), the frame $t-2\Delta t$ (with weight 0.125), and so on. The first time-averaged frame $\bar{F}(j,k,0)$ can be defined equal to the frame $F(j,k,0)$ or as a completely black image. Figure 4 depicts a time-averaged frame. After the time averaging, most of the objects in the frame become blurred, except the logo and perhaps some other time-invariant objects.

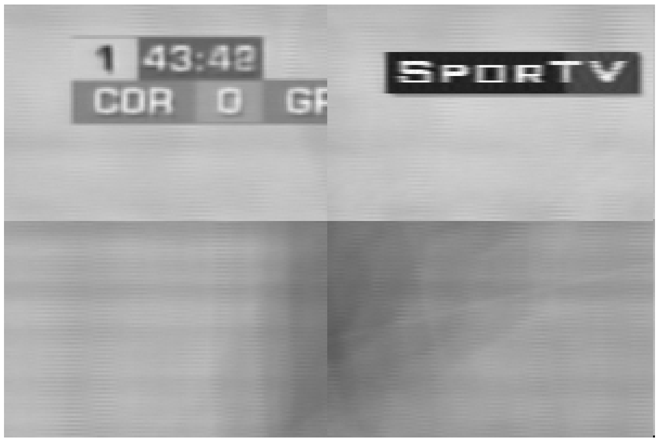


Fig. 4: A time-averaged frame.

Now, the edges can be extracted. There are many different edge-finding methods in the literature. We use the magnitude G of the gradient of the time-averaged image $\bar{F}$ as the edge image:

$$G(x,y) = \left\|\nabla \bar{F}(x,y)\right\| = \left\{\left[\frac{\partial \bar{F}(x,y)}{\partial x}\right]^2 + \left[\frac{\partial \bar{F}(x,y)}{\partial y}\right]^2\right\}^{1/2} \quad (2)$$

The convolution of the time-averaged image $\bar{F}$ with Prewitt operators (figure 5) can be used to evaluate the two partial derivatives of the equation above. Figure 6 depicts some examples of the edge images G.

We have tested also detecting edges before performing the time averaging, and similar results were yielded.

−1	−1	−1
0	0	0
1	1	1

−1	0	1
−1	0	1
−1	0	1

Fig. 5: Prewitt operators

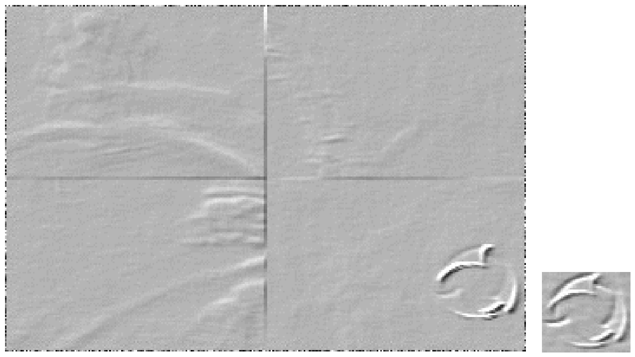


Fig. 6: The edge image G of a video stream and the edge image of a semi-transparent logo.

III. Template Matching

In order to recognize logos in a video stream, it is necessary to have a dataset of edge images of logos, say L_1, L_2, ..., L_n, obtained by pre-processing the sample videos as described in the previous section. Associated with each logo, there must be a list of one or two sub-videos where the logo can appear. Given a video stream, the edge image G of the time-averaged video $\bar{F}$ is computed for each Δt frames.

Then, the cross-correlation is used to spatially localize the logo. Before the correlation, the images are first mean-corrected, that is, the "DC level" is taken out:

$$\tilde{G}(j,k) = G(j,k) - \bar{G}(j,k) \quad (3)$$

$$\tilde{L}_i(j,k) = L_i(j,k) - \overline{L}_i(j,k), \quad 1 \le i \le n \quad (4)$$

where $\overline{G}(j,k)$ and $\overline{L}_i(j,k)$ are the mean grayscale levels of the edge image *G* (at frame *t*) and of the logo image L_i.

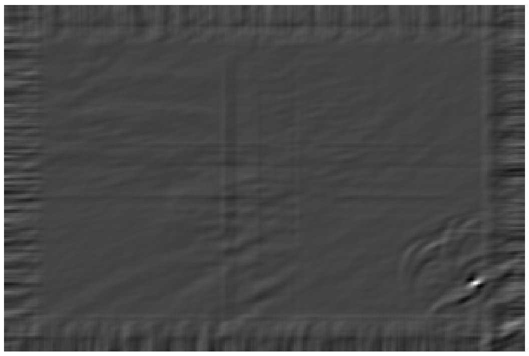


Fig. 7: Convolution of the two images of figure 6. The matching is at the brightest point.

The cross-correlation *R* between two real-valued images $\tilde{F}$ and $\tilde{L}_i$ is defined [8, 9]:

$$R(m,n) = \sum_j \sum_k \tilde{F}(j,k)\tilde{L}_i(j+m,k+n)\,. \quad (5)$$

The cross-correlation can be normalized, by dividing it by the length of vectors $\tilde{F}$ and $\tilde{L}_i$, and yielding the correlation coefficient:

$$\gamma(m,n) = \frac{\sum_j \sum_k \tilde{F}(j,k)\tilde{L}_i(j+m,k+n)}{\left\{\sum_j \sum_k \left[\tilde{F}(j,k)\right]^2 \sum_j \sum_k \left[\tilde{L}_i(j+m,k+n)\right]^2\right\}^{1/2}} \quad (6)$$

The correlation coefficient $\gamma(m,n)$ ranges from -1 to 1. Empirically, we estimated the threshold level 0.73 that did not produce any false alarms and found all logos in our video streams.

A more statistically sound decision can be made by performing a hypothesis test. To test the hypothesis, the cross-correlation is converted into a Student's *t*-statistics τ. Then, the hypothesis "the logo $\tilde{L}_i$ is located in edge image $\tilde{G}$ at pixel (*m*, *n*)" can be statistically tested. The underlying supposition is that the pixel values in $\tilde{G}$ are generated independently at random (this supposition is not completely true).

Let us denote the pixel values of image $\tilde{G}$ scanned in some predefined order (like raster order) as one-dimensional vector *Y*. Similarly, let us denote the pixel values of image $\tilde{L}_i$ translated to position (*m*, *n*) and scanned in the same order as one-dimensional vector *X*. Then, the objective is to estimate parameter β that minimizes error ε in the following equation:

$$\begin{bmatrix} Y_1 \\ \vdots \\ Y_N \end{bmatrix} = \begin{bmatrix} X_1 \\ \vdots \\ X_N \end{bmatrix} \beta + \begin{bmatrix} \varepsilon_1 \\ \vdots \\ \varepsilon_N \end{bmatrix} \quad (7)$$

This equation is usually written in matrix notation as:

$$Y = X\beta + \varepsilon\,. \quad (8)$$

ε is the vector of residual errors, which are considered independent identically distributed normal variables. The parameters β that minimizes the mean square value of error ε can be estimated by the least squares procedure:

$$\beta = \frac{XY}{X^2}\,. \quad (9)$$

The parameter β can be transformed into the Student's *t* statistic τ by computing:

$$\tau = \frac{\beta}{\sqrt{\dfrac{\varepsilon^2}{X^2(n-1)}}}\,. \quad (10)$$

For large *n*, the Student's *t* statistic can be approximated by the normal statistic. The obtained statistic τ is used to perform the hypothesis test. Assuming that the null hypothesis H_0 indicates no correlation between *Y* (the edge image $\tilde{G}$) and *X* (the logo image $\tilde{L}_i$ translated to position (*m*, *n*)), we would like to know how likely is our measure τ. The hypothesis test allows us to perform a comparison between the obtained value τ and the value τ_α corresponding to the selected significance level α (the acceptable false positive rate), accepting or rejecting the null hypothesis if $\tau < \tau_\alpha$ or $\tau \geq \tau_\alpha$, respectively.

The following simple numerical example clarifies these ideas:

$$\begin{bmatrix} -6.375 \\ -5.375 \\ 3.625 \\ 5.625 \\ -5.375 \\ -4.375 \\ 5.625 \\ 6.625 \end{bmatrix} = \begin{bmatrix} -0.5 \\ -0.5 \\ 0.5 \\ 0.5 \\ -0.5 \\ -0.5 \\ 0.5 \\ 0.5 \end{bmatrix} \beta + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \\ \varepsilon_6 \\ \varepsilon_7 \\ \varepsilon_8 \end{bmatrix} \quad (11)$$

The first vector *Y* is the pixel values of the edge image $\tilde{G}$ of the video stream. The second vector *X* is pixel values of the edge image $\tilde{L}_i$ of the logo shifted to position (*m*, *n*). Estimating the parameter β, we obtain 10.75, and estimating the Student's *t* distribution with 7 degrees of freedom we get $\tau = 15.4818$. This means that the logo image was probably found at position (*m*, *n*) of the video stream. The null hypothesis will be rejected at $\alpha = 0.01$ significance level $\tau_\alpha = 3.14$ for a one-tail *t* test. Experimentally, τ assumes values as high as 60 whenever there is a logo matching.

IV. Implementation Details and Experimental Results

A. Implementation in C++

Using the image-processing library called ProEikon [10], we have implemented the proposed algorithm. This implementation does not work in real-time. It was used only to test quickly the ideas developed in the previous sections.

We have used a TV Card named Play TV Pro Ultra by PC View [11] to capture videos from broadcast stations and store them as AVI files.

Ten logos were created from the video files captured from 10 broadcast stations. Five logos were opaque (CNN, SporTV, AXN, Record and Universal), four were semi-transparent (Record, Globo, Gazeta and Cultura) and one partially animated (Discovery).

After creating the dataset with the ten logos, 10 new videos from these 10 broadcast stations and other 5 videos that did not contain any of the 10 logos were captured. All logos were correctly detected, and all absences of logos were also correctly detected. However, it took different times to detect the logos. Opaque logos were detected in average after 1 second. Semi-transparent logos were detected in average after 5 seconds, depending on the variation of the background (the more variation, the less time takes to detect the logo).

B. Implementation in Embedded System (DSP)

We have implemented a complete embedded system environment using DSP (digital signal processor) in C and assembly language. We have used the development kit Blackfin, model EZ-KIT LITE BF533, from Analog Devices [12]. This development board contains all the hardware necessary for this application: volatile flash memory to store the dataset, non-volatile fast memory to compute data, a DSP processor with clock up to 600MHz, a decoder of PAL-M video (Brazilian color TV broadcast standard) and some serial ports to communicate with a PC computer. Figure 8 and 9 depicts our system implemented in a Blackfin board.

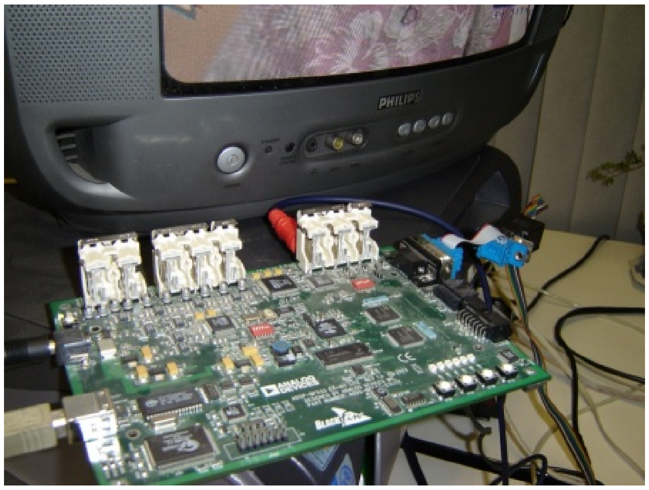
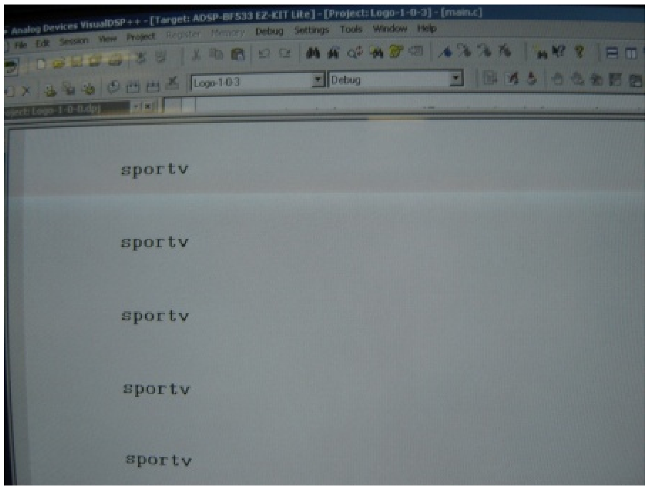


Fig. 8: Our system implemented in Blackfin DSP board.


Analog Devices VisualDSP++ - [Target: ADSP-BF533 EZ-KIT Lite] - [Project: Logo-1-0-3] - [main.c]
File Edit Session View Project Register Memory Debug Settings Tools Window Help
Logo-1-0-3
Debug

```
sportv

sportv

sportv

sportv

sportv
```

Fig. 9: The broadcasting station identified in the PC monitor.

Our system downloads logos from a PC computer, makes acquisition of the video stream in real time, makes video and image processing as described in previous sections, and sends to PC the identity of the detected logo through an asynchronous serial port.

The video's decoder receives a PAL-M signal, and generates a video stream in ITU-656 format [13] in real-time with 8 bits word. In this application we are interested just in Y signals (luminance) that represents gray-scales.

The logo searching is made sequential and exhaustively, logo-by-logo, in the sub-regions specified in the dataset. This process is called "logo detection." When there is a correlation coefficient larger than a specified threshold, the DSP inform it to the PC through a serial port. When a broadcast station is identified, another process called "logo tracking" begins. It consists in confirming this logo until it changes or disappears.

Figures 8 and 9 depict the entire application in our laboratory. The environment consists of a television, the development board and a PC that displays in its monitor the identity of the logo found in the video stream.

We have used the same dataset as the implementation in C++. The same ten logos were used: five opaque logos, four semi-transparent ones, and one partially animated.

After creating the dataset with the 10 logos, we have monitored 15 broadcast stations for more than 24 hours: 10 stations that corresponds to the 10 logos, and 5 stations that do not correspond to the stored logos. All logos were correctly detected without false alarms. However, it took different times to detect the logos. Opaque logos were detected in average after 1 second. Semi-transparent logos were detected in average after 5 seconds, and the time delay depends on the variation of the background (the more variation, the less the time delay).

V. Conclusion

In this paper, we have presented a real-time portable logo detection system. The proposed algorithm is based on edge-based template matching, and requires only small amount of memory and low processing power. It was implemented on a DSP board. All three kinds of logos (opaque, semi-transparent and partially animated) could be detected. We have tested our system for more than 24 hours and all logos were correctly detected.

REFERENCES

[1] A. Divakaran, . Radhakrishnan, "Logo Detection and Classification in a Sport Video: Video Indexing for Sponsorship Revenue Control," Proc. *Int. Soc. Optical Engineering (SPIE)*, pp. 183-193, 2002.

[2] A. Albial, M. J. C. Fulià, A. Albial, and L. Torres, "Detection of TV commercials," *Proc. ICASSP'04,* vol. III, pp. 541-544, May 2004.

[3] J.–H. Yeh, J.–C. Chen, J.–H. Kuo, J.-L. Wu, "TV Commercial Detection in News Program Video," *Proc. Int. Sym. Circuits and Systems (ISCAS)*, vol.5, pp. 4594-4597, May 2005.

[4] W. Q. Yan and M. S. Kankanhalli, "Erasing Video Logos Based on Image Inpainting," in Proc. *Int. Conf. on Multimedia and Expo (ICME),* Switzerland, vol.2, pp. 521-524, Aug. 2002.

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

[6] K. Meisinger, T. Troeger, M. Zeller, and A. Kaup "Automatic TV Logo Removal Using Statistical Based Logo Detection and Frequency Selective Inpainting" *Proc. European Signal Processing Conference*, September 2005.

[7] J. Wang, L. Duan, Z. Li, J. Liu, H. Lu, and J. S. Jin, "A Robust Method for TV Logo Tracking in Video Streams" in *Proc. IEEE Int. Conf. on Multimedia and Expo (ICME)*, pp. 1041-1044, 2006.

[8] K. R. Castleman, "Digital Image Processing," Prentice-Hall, 1996.

[9] R. C. Gonzalez, R. E. Woods, "Digital Image Processing," second edition, Prentice-Hill, 2002.

[10] http://www.lps.usp.br/~hae/software, accessed on September 26, 2006.

[11] http://www.pixelview.com.br/, accessed on August 02, 2006.

[12] http://www.analog.com/processors/platforms, accessed on August 11, 2006.

[13] http://en.wikipedia.org/wiki/ITU656, accessed on August 05, 2006.