Real-time Object Tracking in High-Definition Video Using Frame Segmentation and Background Integral Images

Gustavo Moreira[1], Bruno Feijo[1], Hélio Lopes[1], Raul Queiroz Feitosa[2]
[1]Departamento de Informática
[2]Departamento de Engenharia Elétrica
Pontifícia Universidade Católica do Rio de Janeiro
Rio de Janeiro, RJ, Brazil Email: [1]{gmoreira, bfeijo,lopes}@inf.puc-rio.br; [2]raul@ele.puc-rio.br

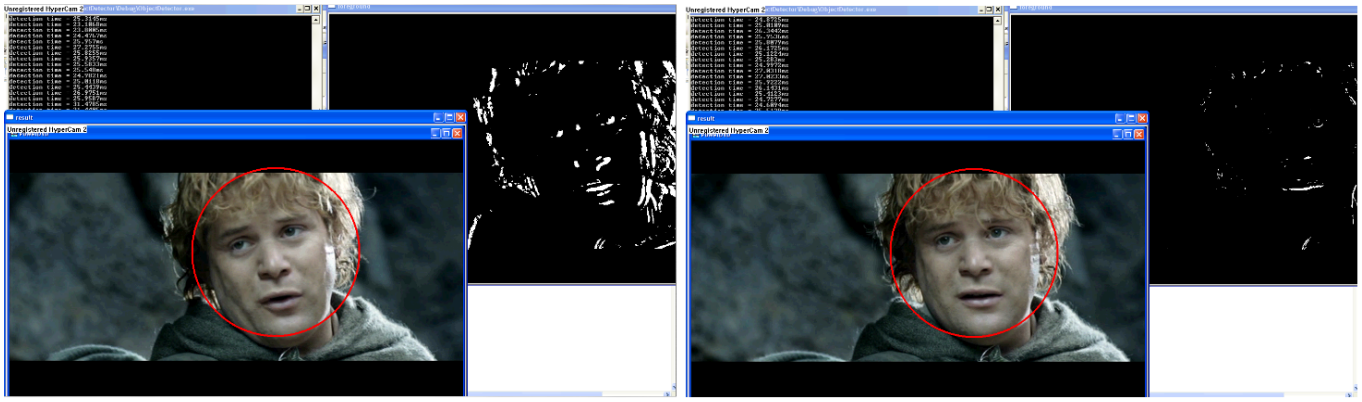


Figure 1. Example of object tracking in two frames of a high-definition video [1] using our algorithm. An usual approach to search for an object is to slide a search window through all the image area. By using frame segmentation and background integral images, we are able to tease out image regions unnecessary for object detection. As a consequence, we achieve high frame rates even in HD videos. Copyrighted images reproduced under "fair use" policy.

***Abstract*—Object tracking in video sequences is still a challenging issue in real-time video processing. In this paper we propose an integrated detection and tracking method suitable for high-definition videos at real-time frame rates. In this method we implement a frame segmentation procedure using integral images of the background, which permits to discard the analysis of several image parts of each frame and achieve high frame rates. Also we extend the proposed algorithm to detect multiple objects in parallel.**

***Keywords*-object tracking; real-time tracking; HD videos.**

I. Introduction

Tracking objects in video sequences is a central issue in many areas, such as surveillance, smart vehicles, human-computer interaction, augmented reality applications, and interactive TV to name but a few. It is a process that always involves two steps: detection and tracking. A common approach is to detect the object in the first frame and then track it through the rest of the video. However, this type of approach overlooks spatial information. A better approach is to pursue a continuous integration of spatial and temporal information, that is: an integrated detection and tracking approach. Examples of integrated approaches can be found in [2]. Another view of tracking methods is the taxonomy found in the survey presented by Yilmaz et al. [3] that comprehends three classes: Point Tracking (objects detected in consecutive frames are represented by points); Kernel Tracking (objects are represented by a kernel, e.g. a rectangular shape or an elliptical shape); Silhouette Tracking (objects are represented by their contour or the region inside a contour). The first class of methods is suitable to handle missing information, and small objects. However, this type of method requires a mechanism to detect the objects in every frame. The second class is suitable for rigid objects, which is appropriate for intensive real-time applications. The third one is suitable for non-rigid and complex objects.

In particular, Kernel Tracking methods essentially use two models: one that gather information from the most recent observation (what may cause the object being tracked to be lost if the object appears different from a different view) [4], and another model where different views of the object can be learned offline and used for tracking [5], [2]. This second model requires training.

Motivation. Our work aims at conceiving a novel method for object tracking that meets the computational performance required for interactive TV programs, where some objects are tracked as hot spots for user interactions (e.g. character selection) or as regions for real-time compositing (e.g. illuminating

a face or attaching an image/text to a character).

The reason why is so difficult to achieve real-time detection and tracking of objects is twofold: firstly, the existence of more than one object to be detected in each video frame compromises the performance of the algorithms; secondly, the use of high resolution videos directly affects the processing time, because the higher is the resolution, the larger will be the area that must be searched for each object.

Contributions. In this paper we propose a real-time integrated detection and kernel tracking method, in which the features of the objects are learned by an adaptive boosting algorithm. Our method is based on the detection method proposed by Viola and Jones [6]. Our first contribution is the use of integral image of the background to discard the analysis of several parts of each frame. Moreover, in the proposed method, the frames are segmented in an adaptive way. Our second contribution is an extension of these ideas to deal with multiple objects in parallel. Our results show a good performance of the proposed method when we deal with high definition resolutions as one can found in digital television videos.

Paper outline. Section II discusses some previous and related works in Kernel Tracking. Section III describes the Viola and Jones' algorithm. Section IV presents our method. Section V shows the experimental results. Finally, Section VI concludes this woe by suggesting future directions.

II. Related Work

For a video sequence with static camera, several authors [7], [8], [9] have tried to develop a robust system for real time tracking. There are also methods of object tracking based on color particle filters [10], [11]. In these methods, the target model of the particle filter is defined by the color information of the tracked object. The tracking of objects is computationally very expensive. There are some techniques for tracking objects that try to reduce the computational cost, such as optical flow, parametric models motion, and matching blocks.

Viola and Jones [6] proposed a reliable algorithm that can detect objects in images in real time, which, according to them, is fifteen times faster than other previously proposed techniques such as [12], [13] and [14]. Lienhart [15] proposed an algorithm based on a set of rotated Haar-like features that enriches the simple features of Viola and Jones' work.

Tresadern et al. [16] proposed a real-time facial feature detection on mobile devices based on integral images.

The great advantage of the Viola and Jones' solution is that it is an algorithm based on features rather than on pixels, what provides much higher performance.

Viola et al. [2] present an extension of Viola and Jones' algorithm [6] to the motion domain. However that extension is focused on low resolution videos of human figures under difficult conditions. Moreover the frame rate is too low (about 4 frames/sec).

III. Viola and Jones' algorithm

As mentioned in the introduction section, the technique presented in this paper extends the Viola and Jones' algorithm [6]. In this section we present a brief description of it.

A. Features of an Object

In Viola and Jones' algorithm, object detection is based on features that differentiate them from other elements in the scene. The main motivation for the use of features rather than pixels for object detection is that it is much faster to compute.

It uses three types of features, as shown in Figure 2. Two "two rectangle features" (Figure 2A and 2B) are given by the difference of pixel values within two rectangular regions. These regions have the same size and are adjacent. A "thee rectangle feature" is given by the sum of pixel values within two rectangles on both side of a central rectangle (Figure 2C). Finally, a "four rectangles" feature is the difference of pixel values within two pairs of rectangles aligned diagonally (Figure 2D).

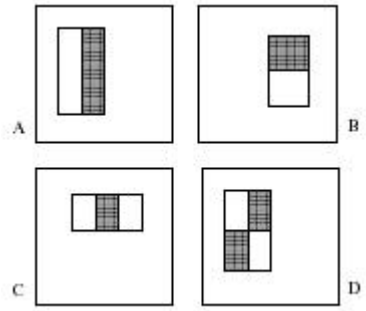


Figure 2. (A) and (B) are two rectangle features; (C) is a three rectangle features; and (D) is a four rectangle features. The sum of white rectangles is subtracted from the sum of dark rectangles. (Figure extracted from [6])

Rectangular features (also called "Haar Features" due to its resemblance to the definition of "Haar Wavelets" [17]) are extremely easy to be computed if we use an intermediate representation of the image called Integral Image. In this form of representation, initially presented by Crow [18], each point (x, y) of the integral image contains the sum of pixels value from the origin to its location, formally:

$$II(x,y) = \sum_{x' \leq x, y' \leq y} I(x', y')$$

Thus, with a single pass on the image, it is possible to compute the integral image. Once the integral image has been computed, we can calculate the sum of pixel values within any rectangular region of the input image reading only four array elements, as illustrated in Figures 3 and 4.

B. Classification Functions

The main idea of Viola and Jones' proposal [6] is to discover a small set of aforementioned features upon which instances of the object can be easily detected by the use of a good classifier. To find out what are these features a classification algorithm known as AdaBoost [19] is used.

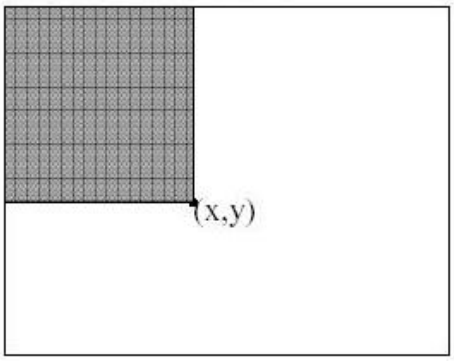


Figure 3. The value of the integral image at point x,y is the sum of all pixels above and to the left. (Figure drawn from [6])

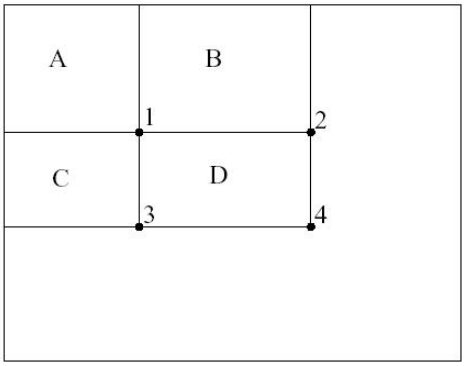


Figure 4. It is possible to find the value of area D calculating: 4+1-(2+3).

In the example of Figure 5, the classifier searches the image for regions, whose features are typical of the object of interest.

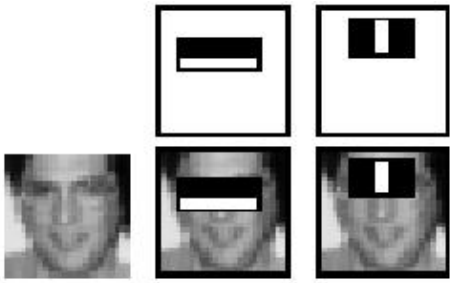


Figure 5. In the first row are examples of the first and second features selected by AdaBoost for training the object "frontal upright human face". On the line below they are superimposed on someone's face. We can see that the first feature shows that the eye region is usually darker than the upper cheek region. The second feature shows the difference in intensity between the region of the eyes and the nasal bridge region. (Figure extracted from [6]).

C. Cascade of Classifiers

Viola and Jones' algorithm uses a cascade of N classifiers to recognize objects. Stages in a cascade are created by ranking functions using AdaBoost [19]. Early stages must discard a large number of image regions that do not contain the desired object, and later stages should be more precise to avoid false positives, as illustrated in Figure 6. If an image region passes through the last stage of a cascade, then this region has the desired object.

It is worth mentioning that the input image processed in different scales. The use of the integral image allows for efficient re-scaling, as will be shown in the following.

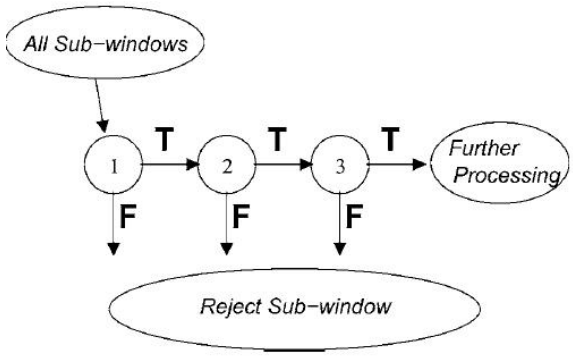


Figure 6. Representation stages of a cascade of classifier. Each stage classifiers (C0, C1, ..., CN) attempts to drop the maximum number of sub windows, in order to decrease the further processing of the sub image, as proposed by [6].

IV. The Proposed Method

This section presents the changes we proposed in the object search algorithm of [6] to enhance its performance, allowing it to remain fast enough for video playback.

A. Search Area Reduction

According to the object detection algorithm originally proposed by Viola and Jones [6], to find a certain object in an image, a new search over the image must be started. This search traverses the entire image by moving a window with a varying size. The window in question begins with the minimum possible size of an object in the image (e.g. 25x25 or 35x40). Each time the window finishes the search process in the whole image, it must be increased in size by a factor λ, and a new search over the image must be started. This procedure ensures that the object that is present in the image is detected regardless of its size. Algorithm 1 summarizes the Viola and Jones' methodology.

We notice here that the entire image area is always tested to check if it contains the desired object. However, it is known that the changes between consecutive frames occur only in few small regions (except when there is a change of scene or a sudden change in lighting, for example). In this case, we can minimize the search area by checking only the regions that have changed as shown in Figure 7.

To accomplish this task, we propose the use of a non-statistical method for the background segmentation by an adaptive mean [20]. The main advantages that led us to use the subtraction technique through adaptive mean are that it is recursive and therefore there is no need to maintain a buffer memory for storing the background model. Furthermore, this technique deals gracefully with changes in lighting and physics, and has a low computational cost.

Algorithm 1 Search Object in Image (Original [6])

Given J, the search window (initialized with the minimum size of the object);
Given I, the original image;
Given λ, the scale factor;
Given Δ, the displacement of the window;
while $size(J) < size(I)$ **do**
  $x = 0;\ y = 0;$
  while $y + \Delta < height(I)$ **do**
    while $x + \Delta < width(I)$ **do**
      if J contains the wanted object **then**
        store the actual location (x, y) of the search window;
      end if
      increment x;
    end while
    increment y;
  end while
  scale the size of J by λ;
end while
Mark in the image the detected locations;

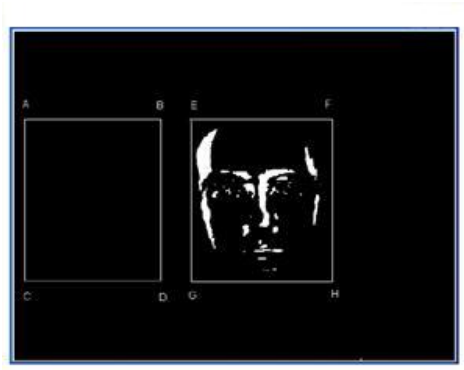


Figure 7. Two rectangular regions on the plane ahead of a time t the video. The rectangle formed by the points A, B, C and D need not be analyzed since the sum of the integral image is zero.

Background segmentation is started by computing a pixel based absolute difference between each incoming frame I_i and a background frame B_i. The background model can be represented by

$$B_{i+1}(x,y) = \begin{cases} I_i(x,y) & \text{if } i = 1 \\ (1-\alpha)B_i(x,y) + \alpha I_i(x,y) & \text{if } i > 1 \end{cases}$$

where α is the learning rate. In addition, the segmentation of objects of interest is given by

$$|I_i(x,y) - B_i(x,y)| > \tau.$$

When the learning rate α is close to zero, the background frame B_i will adapt very slowly to changes in the scene. But when α is close to one, B_i will adapt quickly as objects move and the scene changes.

After the background is subtracted, the search ignores regions with no change in the foreground.

Algorithm 2 Search Object in Image with Segmentation.

Given F, the segmented image;
Given J, the search window (initiallized with the minimum size of the object);
Given I, the original image;
Given λ, the scale factor;
Given Δ, the displacement of the window;

call Segmented_Image_Discarding(F);

function Segmented_Image_Discarding(F')
if $size(F') \leq$ object_min_size **then**
  return object_min_size
end if
divide F' into two parts F_1 and F_2 alternating vertical and horizontal parts on each execution of this function;
compute II_1, the integral image of F_1;
compute II_2, the integral image of F_2;
if $II_1 > 0$ **then**
  ret_size = Segmented_Image_Discarding(F_1);
  call Detect_Object(F_1,ret_size);
end if
if $II_2 > 0$ **then**
  ret_size = Segmented_Image_Discarding(F_2);
  call Detect_Object(F_2,ret_size);
end if
// check if the object is between two regions with changes
// search window doesn't need to be smaller than F_1.
if $II_1 > 0$ **and** $II_2 > 0$ **then**
  call Detect_Object(F',size(F_1));
end if
return size(F')
end function

function Detect_Object (F, search_window_min_size)
scale J until it fits search_window_min_size;
while $size(J) < size(F)$ **do**
  $x = 0;\ y = 0;$
  while $y + \Delta < height(I)$ **do**
    while $x + \Delta < width(I)$ **do**
      if J location on the original Image contains the wanted object **then**
        store the actual location (x, y) of the search window;
      end if
      increment x;
    end while
    increment y;
  end while
  scale the size of J by λ;
end while
Mark in the image the detected locations;
end function

For this test to be done quickly, we firstly set up a binary mask containing the pixels that are part of the foreground. Then we calculate its integral image. Upon the test of whether the window is over an area on which there was a change, we just check whether the integral image area is greater than zero. If so, the window is over an area that may or may not contain the desired object. Otherwise, the area can be discarded from the analysis, because if there was an object and no change occurred in the region of the frame, the object has been found in previous frames. By segmenting the background and calculating the integral images of the segmented parts, we achieve a significant performance gain. To achieve a quick discard of image areas, we propose a "binary search" on each frame, dividing the foreground image recursively in two pieces and analyzing its integral image. If the integral image of the search window is equal to zero, all the area can be discarded, since there is no change between the previous frame and the current one. If the integral image of the current search window is greater than zero, it should be divided and analyzed again until its size is less or equal than the minimal search window size. This idea is illustrated in Figure 8. In addition, Figure shows our program running in a HD video.

Algorithm 2 summarizes our proposal.

B. Parallelism

Since we want to detect multiple objects over a video rather than a single object, we need to optimize the amount of times and the way such objects are tracked in each frame.

The algorithm proposed by Viola and Jones [6] has a solution for searching objects individually. In this algorithm, for each object of interest, it is necessary to rebuild the structures of the cascade classifier, recalculate the integral image, and finally restart the search. In the present paper, we propose a simpler and more effective solution that can be used on videos.

The algorithm can be split in parts that can be executed in parallel, each one dealing with an object instance in the image. These parts are:

- Initialization of the structure of the cascade of classifiers in memory;
- Calculation of the existence of the object in a particular region of the image;
- Marking areas of the image where there are object instances;

We implemented the Algorithm 2 with the above-mentioned parts running in parallel. In this way, we can detect all desired objects simultaneously. This strategy clearly improves the processing time in comparison with the sequential implementation of the Algorithm 1.

V. Experimental results

This section presents the experimental results obtained with different sequences and resolutions of videos. The dataset used for the experiments were known entertainment videos, like Lord of the Rings and The X-Files, and also a pedestrian video, as shown in Figure 9. The following resolutions were used in the experiments: 320 x 240, 640 x 480 and 1280 x 720. The environment used for the tests was an Intel Core 2 Quad 2.4 GHz, 2 GB RAM, Windows XP, and a C++ compiler (Visual Studio 2005) was used as a platform for software development. The Intel OpenCV library was used for the processing and transformation of the original frames to grayscale. A mean value of α = 0.7 was used for the adaptive segmentation of the background. We choose the α parameter empirically. We observe that a very low α value decreases processing time, since the lower the value, the extracted foreground gets bigger and the area to be evaluated increases. A very high value of the alpha parameter does not capture all the necessary changes in the scene, making some objects to not be detected between frames of a video (high number of missing positives). Following the same idea the values of lambda (search window scale factor) and delta (search window displacement) were chosen after empirical experiments.

Table I
AVERAGE PROCESSING TIME IN FRAMES PER SECOND (FPS) SEARCHING A SINGLE OBJECT IN A VIDEO, FOR EACH SPECIFIED RESOLUTION.

Video Resolution	FPS Our Method	FPS Viola Jones
320x240	94.7	54.2
640x480	72.6	28.3
1.024x720	41.3	14.4
1.920x1.080	21.4	5.9

Table II
TOTAL NUMBER OF PROCESSED AREAS LOOKING FOR AN OBJECT ACCORDING TO THE TECHNIQUE ORIGINALLY PROPOSED BY VIOLA / JONES (ALGORITHM I) AND THE PROPOSED TECHNIQUE, USING A SEARCH WINDOW INITIAL SIZE OF 20X20, λ = 1.3 AND Δ = 1. AMOUNTS EXTRACTED IN ACCORDANCE WITH THE FRAME OF FIGURE 7.

Video Resolution	Viola/Jones	Our Method
320x240	81.246	1.984
640x480	162.642	4.206
1.024x720	305.324	7.628
1.920x1.080	620.976	15.842

Our method has the same accuracy of the algorithm originally proposed by Viola and Jones [6]. During our experiments we observed that changing the scale factor λ and the displacement of the search window Δ, we achieved exactly the same number of missing and false positive rates for both methods.

Due to the low computational cost of the proposed method and the high number of discarded areas between frames of a video, it is possible to achieve a high rate of frames per second, including high-definition images, as shown in Table I.

In the experiment of Figure 9, we used four objects of the same class instead of four totally different objects, not only as a matter of implementation convenience, but mainly to avoid the introduction of new variables that could cause a biased analysis of the proposed parallel procedure.

Also we should notice a substantial reduction in the amount of processed areas during the search for objects, as shown in Table II. However, it is important to note that the gain with reduced search area is proportional to the amount of motion

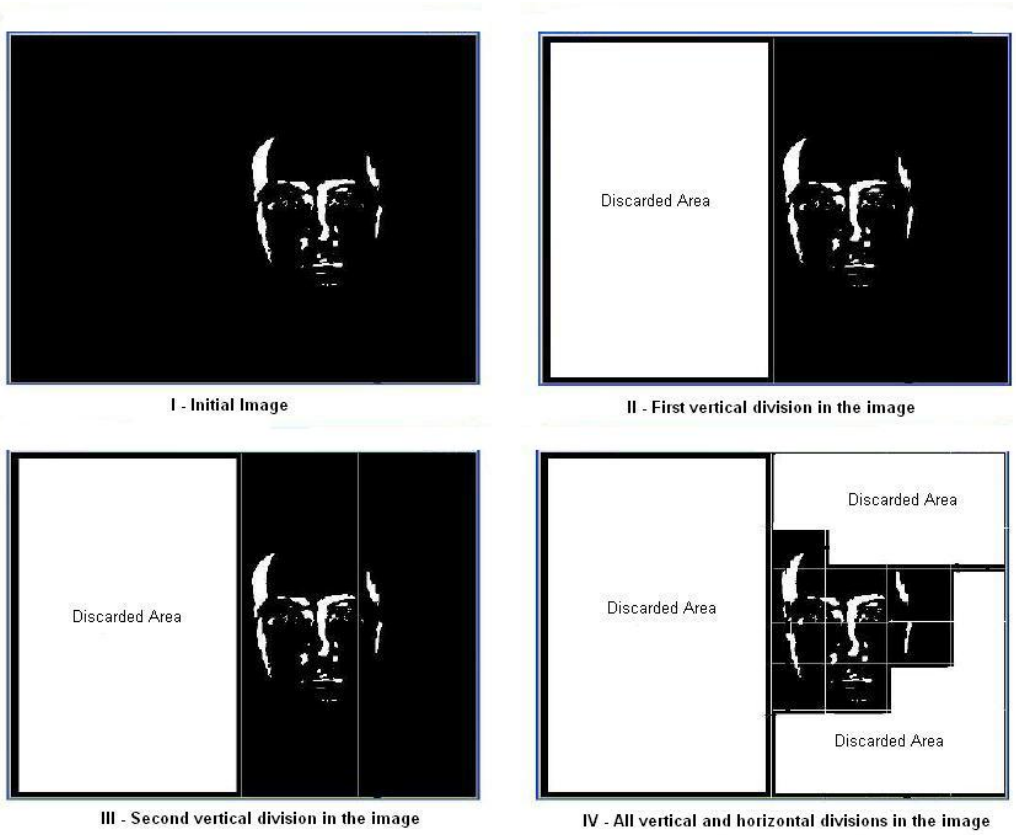


Figure 8. I - It displays the foreground between two frames extracted from a video using adaptive average. II - It shows the first division performed and 50% of the discharge frame according to the respective area, where the sum of the integral image is zero. III - It shows the second division performed on the right side of the image. IV - It displays the effective area for the detection of objects in the frame, where all the white areas are discarded.

between frames. Currently we are not dealing with occlusion, but the method is extensible to detect occluded objects after the end of its occlusion. The processing time obtained by using the proposed parallel technique for multiple objects detection are shown in Figures 10, 11 and 12. In these three figures, we can see that our algorithm produces a quasi-linear behavior with relation to the number of searched objects.

VI. Conclusions

In this paper we extend Viola and Jones' detection algorithm [6] towards a real-time integrated detection and kernel tracking algorithm. Firstly we introduce the idea of using the integral image of the background to discard from the analysis several parts of each frame. Furthermore, this discarding process reveals an adaptive frame segmentation that defines a reduced area for object detection. Another contribution is to expand these ideas to deal with multiple objects in parallel. Finally, our method presents high performance in terms of processing time without missing the qualities of Viola and Jones' original algorithm.

The tests revealed that it is possible to reduce the detection time up to 30% compared to Viola and Jones' method, depending on video resolution and type of object movements. Therefore, the proposed method has the potential to avoid big falls in frame rate when we use high video resolution and search for more than one object.

Our algorithm can reach even higher frame rates if we compute integral images using a parallel all-prefix-sums operation on graphics processors as found in the work by Harris et al. [21]. This operation (also known as a parallel scan) can be applied to all rows of the image followed by the application of the same operation to all columns of the result. Parallel scan is available in CUDA Data Parallel Primitives Library [22]. Also there is the possibility of applying the parallel scan algorithm proposed by Dotsenko et al. [23], [24], which is faster than the one by Harris et al. [21]. We are currently working on this idea. Moreover, as soon as we have this new version of the parallel algorithm, we shall present a complexity analysis - what is missed in the present paper.

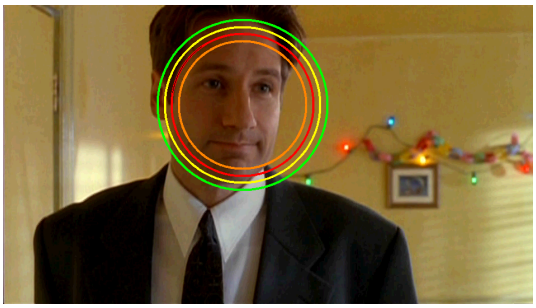
Although there is no widely available testing set in the literature, we have plans to produce more comparisons with published values of performance in terms of frames per second.

The proposed method can facilitate the development of interactive applications synchronized with high resolution digital video in real time. In this particular, the following examples can be mentioned: interactive advertising, e-commerce, or even selling products that are detected during transmissions of TV programs, such as soap operas, reality shows, and movies.

(a)

(b)

(c)



(d)

Figure 9. Examples of object tracking in videos. (a) and (b) illustrates single object tracking (faces) in HD videos. (c) illustrates pedestrian tracking in standard video. (d) illustrates a multi object tracking, in this case 4 objects of the same class "face" are tracked, each one with different radius and color. Copyrighted images reproduced under "fair use" policy [1], [25].

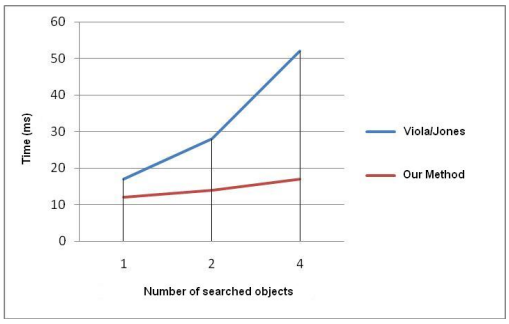


Figure 10. Average performance of the algorithm for detecting objects with and without the proposed optimizations along with a video resolution of 320x240.

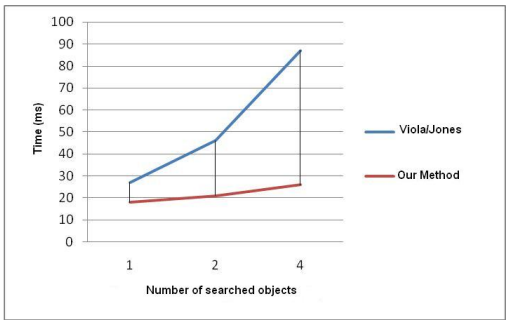


Figure 11. Average performance of the algorithm for detecting objects with and without the proposed optimizations along with a video resolution of 640x480.

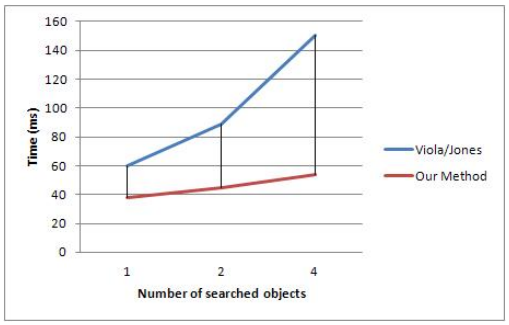


Figure 12. Average performance of the algorithm for detecting objects with and without the proposed optimizations along with a video resolution of 1.024x720.

ACKNOWLEDGEMENTS

The authors would like to thank the reviewers for their valuable comments and suggestions. This work was partially supported by CNPq (National Council for Scientific and Technological Development) and FINEP (Brazilian Innovation Agency), both linked to the Ministry of Science, Technology and Innovation, CAPES (Coordination for the Improvement of Higher Education Personnel, linked to the Ministry of Education), and the Department of Informatics/PUC-Rio.

REFERENCES

[1] N. L. Cinema, "The lord of the rings: The return of the king," 2003.

[2] P. Viola, M. J. Jones, and D. Snow, "Detecting pedestrians using patterns of motion and appearance," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003, pp. 734–741.

[3] A. Yilmaz, O. Javed, and M. Shah, "Object tracking: A survey," *Acm Computing Surveys (CSUR)*, vol. 38, no. 4, p. 13, 2006.

[4] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 24, no. 5, pp. 603–619, 2002.

[5] S. Avidan, "Support vector tracking," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 26, no. 8, pp. 1064–1072, 2004.

[6] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1. IEEE, 2001, pp. I–511.

[7] P. F. Gabriel, J. G. Verly, J. H. Piater, and A. Genon, "The state of the art in multiple object tracking under occlusion in video sequences," in *Advanced Concepts for Intelligent Vision Systems*. Citeseer, 2003, pp. 166–173.

[8] F. Porikli and O. Tuzel, "Human body tracking by adaptive background models and mean-shift analysis," in *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, 2003.

[9] T. P. Chen, D. Budnikov, C. J. Hughes, and Y.-K. Chen, "Computer vision on multi-core processors: Articulated body tracking," in *Multimedia and Expo, 2007 IEEE International Conference on*. IEEE, 2007, pp. 1862–1865.

[10] A. Lehuger, P. Lechat, and P. Perez, "An adaptive mixture color model for robust visual tracking," in *Image Processing, 2006 IEEE International Conference on*. IEEE, 2006, pp. 573–576.

[11] K. Nummiaro, E. Koller-meier, and L. V. Gool, "A color-based particle filter," in *First International Workshop on Generative Model Based Vision*, 2002, pp. 53–60.

[12] R. Cucchiara, C. Grana, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts, and shadows in video streams," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 25, no. 10, pp. 1337–1342, 2003.

[13] N. Friedman and S. Russell, "Image segmentation in video sequences: A probabilistic approach," in *Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence*. Morgan Kaufmann Publishers Inc., 1997, pp. 175–181.

[14] C. P. Papageorgiou, M. Oren, and T. Poggio, "A general framework for object detection," in *Computer Vision, 1998. Sixth International Conference on*. IEEE, 1998, pp. 555–562.

[15] R. Lienhart and J. Maydt, "An extended set of haar-like features for rapid object detection," in *Image Processing. 2002. Proceedings. 2002 International Conference on*, vol. 1. IEEE, 2002, pp. I–900.

[16] P. Tresadern, M. Ionita, and T. Cootes, "Real-time facial feature tracking on a mobile device," *International Journal of Computer Vision*, vol. 96, no. 3, pp. 280–289, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11263-011-0464-9

[17] A. Haar, "Zur theorie der orthogonalen funktionensysteme," *Mathematische Annalen*, vol. 71, no. 1, pp. 38–53, 1911. [Online]. Available: http://dx.doi.org/10.1007/BF01456927

[18] F. C. Crow, "Summed-area tables for texture mapping," *SIGGRAPH Comput. Graph.*, vol. 18, no. 3, pp. 207–212, Jan. 1984. [Online]. Available: http://doi.acm.org/10.1145/964965.808600

[19] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences*, vol. 55, no. 1, pp. 119–139, 1997.

[20] J. Heikkila and O. Silvén, "A real-time system for monitoring of cyclists and pedestrians," in *Visual Surveillance, 1999. Second IEEE Workshop on,(VS'99)*. IEEE, 1999, pp. 74–81.

[21] M. Harris, S. Sengupta, and J. D. Owens, "Parallel prefix sum (scan) with cuda," *GPU gems*, vol. 3, no. 39, pp. 851–876, 2007.

[22] GPGPU.org, "Cudpp – cuda data parallel primitives library," in *Available in http://gpgpu.org/developer/cudpp*, 2013.

[23] Y. Dotsenko, N. K. Govindaraju, P.-P. Sloan, C. Boyd, and J. Manferdelli, "Fast scan algorithms on graphics processors," in *Proceedings of the 22nd annual international conference on Supercomputing*. ACM, 2008, pp. 205–213.

[24] Y. Dotsenko, N. K. Govindaraju, C. Boyd, J. Manferdelli, and P.-P. Sloan, "Matrix-based scans on parallel processors," in *US PATENT 20100076941 A1*, 2010.

[25] T. X.-F. franchise, "The x files," 2002.