FAST TEMPLATE MATCHING FOR MEASURING VISIT FREQUENCIES OF DYNAMIC WEB ADVERTISEMENTS

Dániel Szolgay
Department of Information Technology, Pázmány Péter Catholic University, Práter utca 50/A, Budapest, Hungary
szoda@digitus.itk.ppke.hu

Csaba Benedek and Tamás Szirányi
Distributed Events Analysis Reseach Group, Computer and Automation Research Institute, Kende u. 13-17, Budapest, Hungary
bcsaba@sztaki.hu, sziranyi@sztaki.hu

Keywords: Template matching, web advertisements, integral image, histogram.

Abstract: In this paper an on-line method is proposed for statistical evaluation of dynamic web advertisements via measuring their visit frequencies. To minimize the required user-interaction, the eye movements are tracked by a special eye camera, and the *hits* on advertisements are automatically recognized. The detection step is mapped to a 2D template matching problem, and novel algorithms are developed to significantly decrease the processing time, via excluding quickly most of the false *hit-candidates*. We show that due to the improvements the method runs in *real time* in the context of the selected application. The solution has been validated on *real test data* and quantitave results have been provided to show the gain in recognition rate and processing time versus previous approaches.

1 INTRODUCTION

Analysing human behavior during web browsing is nowadays an important field of research. On one hand, such surveys may provide useful information about abilities, interest or needs of the users, which can be applied in education, offices or e-commerce. We can mention here (Kikuchi et al., 2002), which suggests a method to compare the children's reading abilities in elementary schools.
On the other hand, examining the browsing process helps the web page designers to decide which layouts, colors, font sizes (etc.) may be found ergonomic or attractive by the users. This paper is closely related to this topic: we aim to measure the visit frequencies of embedded web advertisements in given sites. These examinations can help the investors to estimate the efficiency of the commercials, while the web service providers may get information about the optimal locations of the advertisements.
Some corresponding methods in the literature need conscious human feedback: in (Velayathan and Yamada, 2006) the users have been requested to evaluate the web pages they viewed by a questionnaire dialog pop up box. However, in this case the browsing is periodically interrupted by questions, and also ambiguous human factors should be considered such as memory, motivation or preliminary knowledge.
Most of the above artifacts can be avoided if instead of raising queries, we directly measure what the users look at. In the proposed application a special device is used: we track the eye movements of a human observer with the system of the SensoMotoric Instruments® (SensoMotoric, 2005, SMI), called *iView X* (Fig. 1). This system can register the position of the eye with the precision of 0.5 degree in every 50 millisecond.
As for biological background of the inspections, there are three common types of eye movement: fixations, pursuit and saccades. For us the most interesting type of the eye movements is the fixation, because we receive practically all the visual information during it. Thus, after we have got the eye positions from *iView X*, we calculate the coordinates of the fixation points in the screen's coordinate system.
The main task is to implement the automated processing step, which measures the elapsed times meanwhile the users spent in fixation over the different advertisements. The proposed model represents the content of the monitor by screen shots, which are captured periodically. In this way, we can handle the natural user interactions: they may displace the browser

Figure 1: The eye tracker device in use (SensoMotoric, 2005).

window or scroll inside it up and down. Since the advertisements have usually dynamic content (e.g. flash animations), each of them is considered as a sequence of images, which are given at the beginning of the tests. Therefore, the procedure can be considered as a *2D template matching* task from computer vision.
In this paper, a fast template matching method is proposed, which can work in *real time* in the context of the above application. At the end, we validate our solution via *real test data*, and quantitatively show the gain in recognition rate and processing time versus a reference approach.

2 NOTATIONS AND NOTES

Let be N the number of advertisements. Denote by T_{ij} the jth frame (referred later as template) of the ith advertisement. The templates corresponding to the same advertisement are equally sized. The ith advertisement contains n_i frames with size $W_i \times H_i$. The screen coordinates of the current fixation are denoted by $[e_x, e_y]$.[1]
The examined problem can be interpreted as a sequence of 2D template matching tasks in the following way (see Fig. 2): the eye position $E = [e_x, e_y]$ is over the ith advertisement, iff the E-centered, $2W_i \times 2H_i$ sized part of the screenshot contains one of the templates of the ith advertisement ($T_{ij},\ j = 1 \dots n_i$). We will call this $2W_i \times 2H_i$ sized image part as the rectangle of interest: ROI_i^E.
Note that we do not handle cases of partially missing or occluded templates. This problem can be straightforward solved by splitting the advertisement templates into smaller parts and indicating matches if we find at least one part in the search region.

[1]The origin of the screen is in its upper left corner.

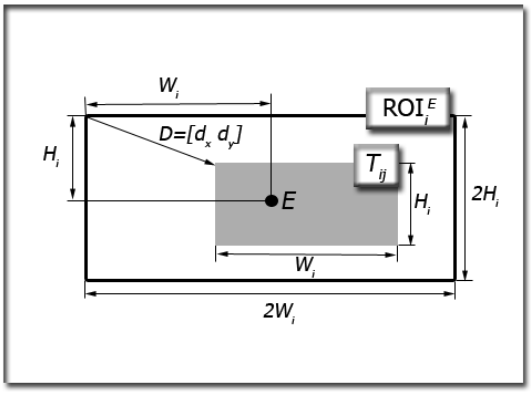

Figure 2: Notations, E: position of the eye; T_{ij} (gray rectangle) a given template; ROI_i^E: $2W_i \times 2H_i$ sized ROI around E.

3 PREVIOUS WORKS

Due to its wide applicability, template matching is a well examined problem. The bottleneck of the available approaches is the time complexity: the general models fail in real time processing of many templates. Therefore, our proposed algorithm highly exploits the application specific a priori conditions such as order of magnitude regarding the size and number of templates. Usually, in a web page (or in a system of corresponding web pages, like a *news portal*) one can observe 1-5 advertisements, however the length of each sequence may be more around hundred frames. Consequently, the advertisement matching procedure should check 100-500 frames in real time. Note that the assumptions about the number of templates only affect the processing time but not the recognition rate of the model, thus the system shows graceful degradation in case of different priors.
For the above reasons, the following descriptions of the different processing steps will focus on their complexity, both regarding our method and the competing models. In most cases, the exact number of the required operations cannot be given, since it also depends on the input images.[2] Thus, we will give the *worst case* values, which already gives a ground for comparison, while the validation will be experimentally performed at the end of the paper using *real data*.

3.1 Basic Method

The basic solution of the template matching problem is the exhaustive search. Let characterize the possible

[2]The models usually consist of a sequence of feature extraction and comparison steps, where the difference between the regions may come forward at any step.

positions of a template T_{ij} by the local coordinates of its upper left corner $D = [d_x, d_y]$ (i.e. displacement) in the coordinate system of the current ROI_i^E. Thus, considering the T_{ij} should be completely included by ROI_i^E (Fig. 2),

$$0 \leq d_x < W_i, \quad 0 \leq d_y < H_i. \tag{1}$$

Pixel by pixel checking of a given template in a given position contains $L_i = W_i \cdot H_i$ comparisons in worst case. For each template, one should check all the possible positions of D row by row, and column by column, thus L_i external cycles are needed. Finally, the above process must be repeated for all templates. In summary, the upper bound of the total number of operations for one screen shot can be calculated as:

$$\sum_{i=1}^{N} n_i \cdot L_i^2. \tag{2}$$

For easier discussion, we introduce the following notations:

$$\overline{n} = \frac{1}{N}\sum_{i=1}^{N} n_i, \quad \overline{L} = \frac{\sum_i n_i L_i}{\sum_i n_i}, \quad \widehat{L} = \sqrt{\frac{\sum_i n_i L_i^2}{\sum_i n_i}},$$

namely, $\overline{n}$ is the average number of the templates of the same advertisement, while $\overline{L}$ ($\widehat{L}$) is the weighted average (quadratic mean) of the template sizes:

$$\min_{i=1\ldots N} L_i \leq \overline{L}, \widehat{L} \leq \max_{i=1\ldots N} L_i.$$

In this way, eq. 2 can be written into the following form:

$$\widehat{L}^2 \cdot \sum_{i=1}^{N} n_i = N \cdot \overline{n} \cdot \widehat{L}^2 \tag{3}$$

Consequently, the number of maximal required operations is proportional to the number of templates and the square of their (quadratic) mean area in pixels. If $\widehat{L} = 120 \times 240$ and we have in aggregate 500 templates this operation number equals to $4.15 \cdot 10^{11}$. This complexity is unadmittable even in case of a few templates.

3.2 Further Approaches

There have been proposed numerous ways to speed up the template matching process. A widely used technique works in the Fourier domain exploiting the phase shift property of correlation (Reddy and Chatterji, 1996). The most expensive part of this procedure is performing the fast Fourier transform which costs for each screen frame and each template $o(L \cdot \log L)$ operations (instead of $\widehat{L}^2$ in eq. 3). Later on, we will see that this performance is still not appropriate. On the other hand, FFT-based techniques can originally register images of the same size, so the smaller template image must be padded e.g. with zeros, which may cause further artifacts. Note as well that due to the current task we do not face rotated templates, which increase the complexity of some general matching models (Yoshimura and Kanade, 1994).

Another approach locates a template within image by histogram comparison (Intel, 2005). As the detailed results of the experimental section show, this solution is also too weak if the number of templates is large.

An obvious way of speeding up the above algorithm can be reached by (spatial) down scaling the templates and the images (i.e. decreasing L_i). However, due to compression artifacts down scaling may cause significant errors, especially in case of line drawings.

A new approximation scheme has been described in (Schweitzer et al., 2002) that requires order $o(L)$ operations for each template. This approach is similar, but more complex than our upcoming "Average feature based filtering" (AFF) step presented in Section 4. Moreover, we will only use AFF for hit validation and exact position estimation, since its complexity is still to high in case of a huge number of templates, which may be present in our application.

Finally, we should mention the iterative template matching techniques (Lucas and Kanade, 1981; Comaniciu et al., 2003), which try to find the optimal location of the template from a given initial position. However, reasonable initializations are needed here, thus these methods are only used, if the positions of the template in the consecutive frames are close to each other (e.g. object tracking with high frame-rate). In our case, the tracker device cannot provide reliable eye positions if the user blinks, thus it is preferred to ignore the high frame-rate assumption.

In the following part of this paper, we consider the number of templates as a constant input parameter, while taking efforts to significantly decrease the average number of the required operations for one template.

4 AVERAGE FEATURE BASED FILTERING (AFF)

We find a key step in the exhaustive search based models, namely, one should quickly answer whether a given template T_{ij} can be placed at a prescribed position D, or not. Since this subroutine is run for all templates and all possible displacement positions, the time complexity of this procedure is crucial. The

above mentioned models needed in average $\widehat{L}^2$ and respectively $o(L \cdot \log L)$ operations at each template. The following algorithm reduces it to $C \cdot \overline{L}$ where C is a small constant.

The main idea in the following approach is that we do not compare pixel by pixel the templates and the template-candidate ROI-parts, but we only compare a few features which can be extracted very quickly. Denote the number of features by F.

First of all, we convert the images into the HSV color space, which is close to the physiological perception of human eye. The features will be the empirical mean values (M) and the mean squared values (Mq, second moment) over the H and V image channels. Thus, each template will be featured by $F = 4$ constants, namely M_{ij}^H, Mq_{ij}^H, M_{ij}^V, Mq_{ij}^V, which should be calculated only once when the program is initialized.

The features over the rectangular hit candidate regions of the ROIs should be calculated in each processing step. Since the templates of a given advertisement are assumed to be equally sized, the examinations must be independently performed for each advertisement but not for the included templates. Given ROI_i^E one should calculate the F feature scalars for each possible template offset $D = [d_x, d_y]$ over the intervals defined by eq. 1. In a naive way, every such calculation would consist of an averaging over a $W_i \times H_i$ search region, which contains $W_i \cdot H_i - 1$ addition operations. Fortunately, the procedure can be significantly speeded up with the widely used integral trick.

4.1 Feature Extraction with the Integral Trick

Several features over a given rectangular neighborhood can be computed very rapidly using an intermediate representation for the image which is called the integral image (Viola and Jones, 2001).

Given an image Λ, its integral image I_Λ is defined as:

$$I_\Lambda(x,y) = \sum_{i=1}^{x} \sum_{j=1}^{y} \Lambda(i,j).$$

With notation $\varkappa(x,0) = 0$ and $I_\Lambda(0,y) = 0$, $x = 1 \ldots S_x$, $y = 1 \ldots S_y$:

$$\varkappa(x,y) = \varkappa(x,y-1) + \Lambda(x,y),$$

$$I_\Lambda(x,y) = I_\Lambda(x-1,y) + \varkappa(x,y),$$

the integral image can be computed in one pass over the original image (with two additions at each pixel).

With the integral trick, the sum of the pixel values over a rectangular window can be computed via three additional operations independently from the window size $(c-a) \times (d-b)$:

$$\sum_{i=a}^{c} \sum_{j=b}^{d} \Lambda(i,j) = I_\Lambda(c,d) - I_\Lambda(a-1,d) - $$

$$- I_\Lambda(c,b-1) + I_\Lambda(a-1,b-1).$$

If we need the sum of the squares of the elements in a rectangular window, the procedure is similar, but a different integral image should be built up for the squared elements.

Consequently, in the current procedure we should calculate F integral images for each advertisement which cost $F \cdot 2W_iH_i$ additions. Thereafter, at each of the possible W_iH_i positions of displacement D, we can calculate the local feature value with 3 additions and 1 multiplication, which should be compared to the reference features of the templates.

4.2 Optimizing the Comparison Via Binary Search

After the algorithm calculates the local first and second ordered means regarding a given D displacement position in ROI_i^E, the resulting features should be compared to the pre-calculated values regarding all the templates of the ith advertisement. This means n_i comparisons. However, if we preliminary put the templates into F ordered lists according to the F calculated feature values, $F \cdot (1 + \log_2 n_i)$ comparisons are enough using a binary search. In summary, the time complexity of the method is as follows (henceforward using $L_i = W_iH_i$):

$$\underbrace{2F \sum_{i=1}^{N} L_i}_{\text{integ. im.}} + \underbrace{4F \sum_{i=1}^{N} L_i}_{\text{feature calc.}} + \underbrace{\sum_{i=1}^{N} L_i \cdot (1 + \log_2 n_i)}_{\text{compare}}$$

This is much more promising than the complexity defined by eq. 2, and also better than using the FFT-based model. In our example, with $N = 4$, $F = 4$, $L_i = 120 \times 240$ and $n_i = 100$ the number of required operations is $3.64 \cdot 10^6$. However, according to our tests this speed is still not enough for real time processing.

5 HISTOGRAM BASED FILTERING (HF)

Although the complexity of the AFF step is too high in case of a huge number of templates, it can efficiently validate a few hit candidates. In this section, we present a quick preprocessing algorithm, which

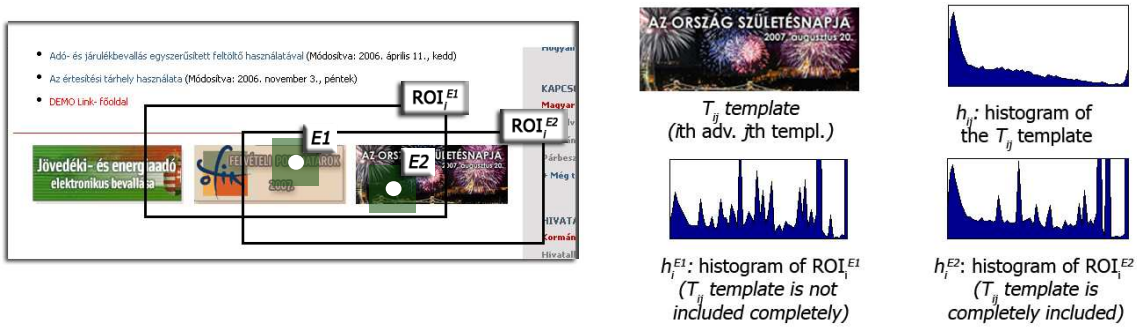

Figure 3: Demonstration of the histogram based filtering step. Let be $E1$ and $E2$ two potential eye positions, and we are looking for a given template corresponding to the ith advertisements as shown below. ROI_i^{E1} and ROI_i^{E2} are the rectangles of interest around $E1$ and $E2$ respectively. Comparing the V histograms shows that h_{ij} is below h_i^{E2} over the whole domain, but this condition does not hold for h_i^{E1}. Thus, only $E2$ should be henceforward examined.

can practically filter out most of the false hits, thus it enables a real time performance.

Denote by h_{ij} the histogram of the T_{ij} template, which uses K bins. Here, $h_{ij}[k]$ denotes the kth bin. Similarly, denote by h_i^E the histogram of ROI_i^E. If ROI_i^E contains the T_{ij} template, the following constraint must hold:

$$h_{ij}[k] \leq h_i^E[k] \quad \text{for all } k = 0 \ldots K-1.$$

Thus, all the templates can be excluded from further examinations, which fail at least one of the above inequalities.

As for computation complexity: h_{ij} should be calculated only once in the program. h_i^E should be calculated at each eye movement for all advertisements (but their number is much lower than the number of templates), it takes L_i operations. For all templates one should only compare the two histograms which needs K operations (we used $K = 64$). Thus the total number of required operations is:

$$\sum_{i=1}^{N} L_i + K \cdot \sum_{i=1}^{N} n_i.$$

The significant gain in running time comes from the fact that the time complexity of the above step depends linearly both on the size and number of templates. With parameters $N = 4$, $\overline{L} = 120 \times 240$, $K = 64$ and $\sum_i n_i = 400$, the number of required operations is $1.4 \cdot 10^5$.

In summary, our procedure begins with performing the quick HF step, which returns a set of template candidates. Thereafter, AFF should be only run for these candidates. Finally, the AFF-hits are validated by a single pixel by pixel comparison to the screen shot, which can completely filter out the false positive matches.

Table 1: Parameters of the test browsing sequences. N: number of advertisements $\overline{n}$: average number of templates belonging to one advertisement *Hits:*. shows how many times the user looked at the advertisements. *Adv. area:* average number of pixels belonging to the advertisements on one frame of the sequence.

	N	$\overline{n}$	Hits	Adv. area
Seq. 1	3	73	43	100860
Seq. 2	3	115	77	113520
Seq. 3	3	1	34	12144
Seq. 4	4	81	51	184810
Seq. 5	3	46	23	147340

6 EXPERIMENTS

The proposed algorithm was evaluated on 5 different browsing sequences, which were captured from different web sites including different advertisements. The test parameters are as follows: the resolution of the screen shots is 1024×768 and each examined web page contains $N = 3$ or 4 dynamic advertisements. During the surveys, there were altogether 228 "eye hits" on advertisements. Further detailed information about the sequences is provided in Table 1.

The results got by our model were compared to an efficient and general template matching technique (Intel, 2005) (see notes in Section 3.2). To examine the effects of downscaling on speed and accuracy, we have also tested the reference method by decreasing the resolution of the screen shot image and the templates. More precisely, we tested four different scaling coefficients: 1.0, 0.5, 0.25 and 0.1.

We measured three evaluation factors in the tests: number of false negative respectively false positive hits (Table 2), and computational time (Table 3). As the results show, the reference method works reliably

Table 2: False negative (FN) and false positive (FP) hits of the reference template matching (TM) algorithm in 4 different scales, and the proposed method on the five testsequences.

		Seq1		**Seq2**		**Seq3**		**Seq4**		**Seq5**	
Method	**Scale**	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP
TM	0.1	22	12	31	13	14	5	8	50	0	1
TM	0.25	20	6	28	6	0	0	25	15	0	1
TM	0.5	34	2	28	4	0	0	6	11	0	1
TM	1.0	0	0	0	0	0	0	0	0	0	0
Prop. Alg.	1.0	0	0	0	0	1	0	1	0	0	0

Table 3: Computational time (in sec) for one fixation point for the reference template matching (TM) algorithm in 4 different scales, and the proposed method (PM) on the five testsequences, using C++ implementation and a Pentium laptop (Intel(R) Core(TM)2 CPU, 2GHz).

		Processing time				
Me.	**SF**	Seq1	Seq2	Seq3	Seq4	Seq5
TM	0.1	0.32	0.42	0.05	0.60	0.28
TM	0.25	2.55	4.42	0.05	4.56	2.29
TM	0.5	9.17	21.8	0.11	22.9	9.33
TM	1.0	> 45	> 45	0.27	>45	> 45
PM	1.0	0.14	0.15	0.08	0.23	0.19

with reasonable speed having only a few templates (Seq. 3), but as the template number grows the computational time dramatically increases (Seq. 4). Although the reference algorithm can run in real time if the input data is downscaled to one tenth of the original resolution, in that case the number of false positive and false negative hits is extremely high. With less drastic down scale the method is more accurate but the computational time highly increases so the process is not real time any more. On the other hand, the proposed algorithm runs in real time (which means that the processing of a fixation point is finished befor the next data arrives) with high accuracy on each sequence even with the highest template number.

7 CONCLUSIONS

In this paper we presented an on-line method for statistical evaluation of dynamic web advertisements by tracking the users' eye movements. For registering the eye movement a special camera was used. The events when the eye "visits" an advertisement were automatically detected with a quick template matching algorithm. The proposed method was tested on *real data* and found to be very accurate (less then 1% error) and fast (*real time*) even in case of a high template number.

ACKNOWLEDGEMENTS

The authors are grateful to Zoltán Vidnyánszky, Viktor Gál, and Márton Fernezelyi for providing the test data. This work was supported by the C3 (Center for Culture and Communication) Foundation, Hungary.

REFERENCES

Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *IEEE Trans. Pattern Anal. Mach. Intell.*, 25(5):564–575.

Intel (2005). *Documentation of the Intel Open Source Computer Vision Library (OpenCV), http://www.intel.com/technology/computing/opencv/*.

Kikuchi, H., Kato, H., and Akahori, K. (2002). Analysis of children's web browsing process: Ict education in elementary schools. In *Proc. ICCE*, page 253, Washington, DC, USA. IEEE Computer Society.

Lucas, B. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *Proc. of International Joint Conference on Artificial Intelligence*, pages 674–679, Vancouver, BC, Canada.

Reddy, B. and Chatterji, B. (1996). An FFT-based technique for translation, rotation and scale-invariant image registration. *IEEE Trans. on Image Processing*, 5(8):1266–1271.

Schweitzer, H., Bell, J. W., and Wu, F. (2002). Very fast template matching. In *Proc. ECCV*, pages 358–372, London, UK. Springer-Verlag.

SensoMotoric (2005). *Documentation of the iView X System, http://www.smi.de/iv/*.

Velayathan, G. and Yamada, S. (2006). Behavior-based web page evaluation. In *Proc. WWW*, pages 841–842, New York, NY, USA. ACM Press.

Viola, P. and Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. In *Proc. IEEE CVPR*, volume 1, pages 511–518, Hawaii, USA.

Yoshimura, S. and Kanade, T. (1994). Fast template matching based on the normalized correlation by using multiresolution eigenimages. In *Proc. IROS*, volume 3, pages 2086 – 2093.