# Fast semantic region analysis for surveillance & video databases

S. Javanbakhti, S. Zinger, P. H. N. de With
Eindhoven University of Technology, Eindhoven, Netherlands
S.Javanbakhti@tue.nl, S.Zinger@tue.nl, P.H.N.de.With@tue.nl

***Abstract*—Video databases are broadly applied both in consumer and professional domains. The importance of real-time surveillance video monitoring has increased for security reasons, while also for consumers video databases are rapidly growing. Quickly searching in databases is facilitated by region analysis, as it provides an indication for the contents of the video. For real-time and cost-efficient implementations, it is important to develop algorithms with low computational complexity. In this paper, we analyze the complexity of a newly develop semantic region labeling approach [2], e.g. road, sky, etc., which aims at extracting spatial contextual information from a video. In the analyzed semantic region labeling approach, color and texture features are combined with the vertical position to label the key regions. The algorithm is analyzed by its native *DSP* computations and memory usage to prove its practical feasibility. The analysis results show that the system has a low complexity while offering high-accuracy region labeling. A comparison with the state-of-the-art algorithm convincingly reveals that our system outperforms the state-of-the-art with fewer computations.**

## I. INTRODUCTION

In today's society with increased security concerns, surveillance video monitoring is broadly applied and video data are stored. Similarly, consumers have started to capture more images and videos than before, because of their steady mobile phone usage. This results in continuously growing databases with pictures and videos. It is evident that large databases require user-friendly access and browsing. This motivates research on fast content-classification algorithms.

Automatic video scene understanding can be worked out with the use of region labeling, if the labels are semantically meaningful (e.g., road, sky, etc.). One dominant application for scene understanding and region labeling is classifying (video) pictures in a large database. Additionally, in surveillance, the region labeling and video understanding can improve the analysis of events or contribute to more reliable object detection.

For a reliable model of a scene and associated context information, the labeling task involves image feature analysis at global and local scene levels. Although local features such as color and texture per pixel or region are instrumental for understanding, they are typically not uniquely determining the semantic meaning of such a region (e.g. sky and water). Furthermore, taking contextual information into account may increase the performance of region labeling [1].

In this paper we propose an automatic region labeling system (see Fig. 1) based on the observation that each region is more likely to be found at a specific vertical image position (spatial context). Besides feature extraction using color, texture and spatial context information, the algorithm exploits scene information in a learning fashion using Support Vector Machine (SVM). The presented system has been published in [2], where more details can be found.

The focus in this paper is on estimating the algorithm complexity to show that the algorithm is not only offering accurate region analysis, but also executes with low complexity, to support real-time and embedded systems.

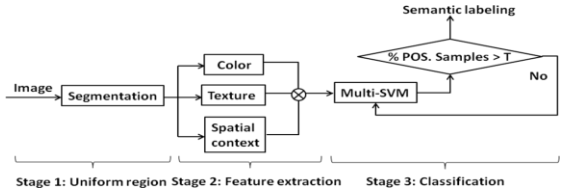


Fig. 1. Our semantic region labeling approach.

## II. COMPLEXITY ANALYSIS METHOD

A known metric for estimating the computational complexity is Mega Operations per Frame (MOPF) or per second (MOPS) [3]. This metric is based on counting native DSP operations, like multiplications, additions, data storing and loading. For example, consider a simple linear equation $y = ax + bz$ where $a$, $b$ are constants and $x$, $y$ and $z$ are data. This requires 4 data loads, 2 multiplications, 1 add, 1 data store, thus 8 operations. By also incorporating data stores and loads, also an indication of the memory usage and bandwidth is obtained.

Summarizing, in the above method a simple RISC CPU processor is assumed with sufficient background memory for video data storage, but with a limited set of registers. In this model, address computations are omitted (this is existing parallelism in processor hardware). Also, the data traffic from cache to memory is omitted (cache hierarchy is present and working properly). In [3], caching was incorporated, but skipped here for time limitations.

## III. ANALYSIS OF REGION LABELING ALGORITHM

This section discusses details of computational complexity calculations of the first stage only (Uniform Region) of our region labeling approach in Fig. 1 because it involves more complicated calculations compared to the other stages. The complete analysis has been performed and will be summarized. Computational complexity calculations of Feature extraction and Classification stages have been obtained in the same way and will be presented in the full paper. In the Uniform region stage, a graph-based segmentation approach is applied, which is motivated by the concept that pixels within one region are closer in color space than pixels from different regions. Based on this, a threshold is defined depending on the size of the region, where it is assumed that larger regions should have larger tolerances. In the graph-based segmentation, the input image is first filtered with a Gaussian kernel (Table I). It is proven that an efficient way to convolve an image with a 2D Gaussian kernel is to separate the Gaussian kernel into two 1D

kernels. The 1D convolution is then specified by $(I*K)(i) = \Sigma K(i).I(i-j+1)$, where $I(i)$ denotes the intensity of the current pixel, $K$ is the 1D kernel (of size 1x2), $i$ and $j$ are the current pixel positions. Since the kernel size is 1x2, the filter needs 2 kernel reads for th1D kernel $K(i)$, 2 image pixel reads for $I(i-j+1)$, 2 multiply accumulations for $\Sigma K(i).I(i-j+1)$ and 1 data store of $(I*K)(i)$ in the memory. Later, this computation is repeated horizontally and further for each channel per pixel.

After smoothing the input image with a Gaussian kernel, the segmentation algorithm requires the steps of region comparison, threshold calculation, threshold comparison and threshold updating (Table I). For further details of the algorithm steps, the reader is referred to [2]. The step of region comparison ($If\ Region_A \neq Region_B$), involves 2 reads for $Region_A$ and $Region_B$ and 1 comparison. We repeat this computation for each channel and per pixel. The step of threshold calculation ($T_A = W_A + k / Size_A$) involves 1 data read of intra-region weight $W_A$, where $W_A$ is defined as the maximum edge weight within $Region_A$. The step of threshold calculations further involves 1 read for the initial inner threshold $k$ which is a tunable input parameter, 1 read for $Size_A$, 1 add, 1 multiplication for $k/Size_A$ and 1 write for the threshold $T_A$. It should be noted that $W_A$ and $Size_A$ are saved in a lookup table in an offline calculation and they are updated later, if applicable and when required conditions are satisfied. The step of the threshold comparison ($T_{A,B} \leq \min(T_B, T_A)$) involves calculating $T_{A,B} = W_{A,B} + k / Size_{A,B}$, where the subscripts "$_{A,B}$" denote taking regions $A$ and $B$ together. Weight $W_{A,B}$ requires 2 reads for $W_A$ and $W_B$, 1 comparison to achieve the maximum of $W_A$ and $W_B$ and 1 write for $W_{A,B}$, then 2 reads for $Size_A$ and $Size_B$, 1 add to obtain $Size_{A,B}$ and 1 add for $Size_{A,B}$. Furthermore, the step requires 1 read for $k$, 1 add, 1 multiplication for $k / Size_{A,B}$ and 1 write for $T_{A,B}$. In addition, the step involves $\min(T_B,T_A)$ which requires 2 reads for $T_B$ and $T_A$ and 1 comparison. Also, the step requires 1 comparison for $T_{A,B} \leq \min(T_B, T_A)$. If the criterion is satisfied, then the algorithm continues to the next step which is updating the threshold. This threshold updating ($T_{A,B} = W_{A,B} + k / Size_{A,B}$) is based on the threshold comparison result. If ($T_{A,B} \leq \min(T_B, T_A)$) holds, then two regions are merged and $W_{A,B}$ and $T_{A,B}$ are updated for two regions. Except for the Gaussian filter (a preprocessing step), all above-mentioned steps are repeated by the height of the graph, which equals $n\ log(n)$ ($n$ is pixels).

## IV. ESTIMATION RESULT

Table I shows the estimation results of the computational complexity of the semantic region labeling on the "LableMe" dataset [4] with the basic frame size of 256 x 256 pixels. In this paper, this public dataset has been selected for scientific reference only. For applications requiring full-HD resolution video, the herein presented calculations should be multiplied by about 32 times and the frame rate, so that MOPF counts become GOPS with nearly the same numbers, proving that region labelling is feasible for real systems. Table I shows that the total operations count for all 3 stages is 24.77 MOPF. For scientific comparison, we also calculated the computational complexity of the semantic region labeling approach of Millet *et al*. [1], which equals 33.6 MOPF (details in the full paper). This comparison shows that our approach outperforms Millet's algorithm with 20% lower complexity, while it was reported in [2] that the region labelling accuracy of our algorithm is at the same time higher than that of Millet *et al*. [1] (93% vs. 90%).

TABLE I
THE COMPUTATIONAL COMPLEXITY OF SEMANTIC REGION LABELING

| Algorithm steps | Formula | Operations/Frame |
|---|---|---|
| Stage 1: Segmentation: | | |
| 1) Gaussian filter | $(I*K)(i) = \Sigma g(i).f(i-j+1)$ | 7x3x2x256x256 |
| 2) Region comparison | If $Region_A \neq Region_B$ | +3x256x log256 |
| 3) Threshold calculation | $T_A = W_A + k / Size_A$ | +1x256x log256 |
| 4) Comparison | $T_{A,B} \leq \min(T_B, T_A)$ | 12x256xlog256 |
| 5) Updating threshold | $T_{A,B}$ | +256xlog256 |
| Stage 2: Color: RGB to HSV | $V = \max(R, G, B)$<br>$S = V - \min(R,G,B)$<br>$H = (G-B) / S$ if $V = R$<br>$H = 2 + (B-R) / S$ if $V = G$<br>$H = 4 + (R-G) / S$ if $V = B$ | 38x256x256 |
| Stage 2: Texture: | $0.299.R+0.587.G+0.114.B$ | |
| 1) RGB2GRY | $(X_1+jY_1).(X_2+jY_2)$ | 9x256x256 |
| 2) Filter construction | $R = \text{imageFFT} * F(f)$ | + 2x7168 + |
| 3) Multiplication | $\text{Ifft2}(R)$ | 26x256x256 |
| 4) Inverse transform | | |
| Stage 2: Spatial context: Normalized position | Vertical $_{position}$/ image $_{height}$ | 100x6x2 |
| Stage 3: Classification: | $\left(\sum_{i \in SV} \alpha_i * k(X_i, z)\right) + N_{SV} * b$ | 12x5x337406+ |
| 1) Stage Kernel function | | |
| 2) Probability | % positive samples | 202 + |
| 3) Maximum probability | Maximum | 201 |
| 4) Thresholding | Maximum > Threshold | 12 |
| Total (MOPF) | | 27.83 MOPF |

## V. CONCLUSION

This paper has evaluated the complexity of our semantic region labeling approach [2] for real-time implementation in (embedded) video data systems. The proposed fast semantic region labeling approach, is based on combining color, texture and vertical position context as feature information in a learning-based fashion. The performed complexity analysis method is based on counting native DSP operations with a basic RISC CPU as a reference model. It was shown showed that our approach outperforms state-of-the art approach of Millet [1] with 20% lower complexity while maintaining higher accuracy.

The full paper will present more details of the complexity analysis and the state-of-the-art approach. Also, we will perform detailed computational complexity analysis of our Security Salient region detection approach and a relevant state-of-the-art approach to evaluate their suitability for real-time implementation in embedded video applications as found in consumer and surveillance.

## VI. REFERENCES


[1] C. Millet, *et al*., "Using relative spatial relationships to improve individual region recognition", EWIMT, 2005.

[2] S. Javanbakhti, *et al*., "Context-based region labeling for event detection in surveillance video", ISEEE, 2014.

[3] R. Albers, "Modeling and control of image processing for interventional X-ray", PhD thesis, Eindhoven University of Technology, 2010.

[4] B. C. Russell, A. Torralba, K. P. Murphy and W. T. Freeman, LabelMe: a database and web-based tool for image annotation, International Journal of Computer Vision, pp. 157-173, vol. 77, 2008.