Automatic Video-Object Segmentation Employing Multi-Sprites with Constrained Delay

Dirk Farin[1] and Peter H.N. de With[1,2]
[1]University of Technology Eindhoven, 5600 MB Eindhoven, Netherlands
[2]LogicaCMG, TSE, 5605 JB Eindhoven, Netherlands

Abstract

This paper proposes an automatic video-object segmentation system for consumer media, based on the background subtraction technique. In order to allow for camera motion, the input images are aligned to a large panoramic background-sprite image. A multi-sprite technique is applied to minimize the size of the synthesized sprite images. In contrast to previous algorithms, which required multiple passes over the input data, the proposed algorithm enables online processing with a fixed processing delay. This is important for implementation in consumer devices which have to run in real-time with constrained resources. We provide example results illustrating that a real-time segmentation on memory-constrained hardware is feasible.

Introduction

Automatic video-object segmentation is a prerequisite for many advanced video analysis and compression techniques. These automatic video analysis systems become increasingly important for consumer products because they enable technologies like intelligent searching in video collections, or home-surveillance systems. Moreover, object-oriented video coding tools like proposed in MPEG-4 yield a higher compression ratio, because they transmit the scene background independently from the foreground objects [4].

The proposed automatic segmentation system is based on the background subtraction technique [1]. The central step in this technique is to synthesize a camera-motion compensated background image (*sprite*) that does not show any foreground objects. Generally, this background image will be much larger than the input resolution if camera motion is present. Foreground objects are removed from the background image by a temporal averaging process. Subsequently, the foreground objects can be found by detecting the change between the background image and the input.

Many television cameras and most surveillance cameras are pan-tilt-zoom cameras, which can rotate around two axes and which can change the zoom. In this case, it is possible to align the images of a camera pan into a sprite image. However, if the camera rotation angle exceeds a specific angle, the area in the sprite onto which the input images are projected grows quickly and, ultimately, it becomes impossible to synthesize a common background image. This restriction can be eliminated by first splitting the input sequence into several frame ranges, and computing independent background sprites for each range. An algorithm to compute the optimal partitioning of the input sequence, minimizing the total sprite area has been proposed earlier [2]. To compute a globally optimal solution, this algorithm has to consider the complete input sequence at once. In a real-time system, this is impossible since online processing with a maximum delay has to be assured.

Unfortunately, a general low-delay video segmentation system is not possible because the sprite image first has to be synthesized in the encoder from many input frames. These processing steps require buffers that store the frames of at least one *segment*, i.e., the frames that will be composed into a single sprite image. Hence, we propose a compromise, in which the maximum delay is constrained by imposing a limit on the number of frames that are combined into one sprite image.

Online Sequence Partitioning

The multi-sprite partitioning splits the input sequence into separate segments, for which separate background sprites will be constructed in the successive steps. In order to reduce transmission bandwidth and computation time in the subsequent processing steps, we desire to minimize the total area covered by all sprites. We first consider the problem of determining the global optimum and then incorporate constraints to limit the maximum and minimum number of frames per segment to fix the processing delay.

Let $P = \big((1, p_1 - 1), (p_1, p_2 - 1), \ldots, (p_{n-1}, N)\big)$ be a partitioning of the video sequence of length N into n segments. The optimization problem can then be formulated as determining the partitioning P^* for which the sum of all sprite costs (required area) is minimal:

$$P^* = \arg\min_P \sum_{(i,k)\in P} ||S_{i;k}||. \tag{1}$$

With $||S_{i;k}||$, we denote the cost for synthesizing a sprite covering frames i to k. This minimization problem can be viewed as a minimum-cost path search in a graph, where the graph nodes correspond to the input frames plus an additional dummy start node, $V = \{0, \ldots, N\}$. The graph is fully connected with directed edges $E = \{(i;k) \mid i, k \in V; i < k\}$. Each edge $(i;k)$ is attributed with edge costs $||S_{i+1;k}||$. Every path from the start node 0 to node N defines a possible partitioning, where each edge on the path corresponds to one segment. Consequently, the minimum cost path gives the minimum cost partitioning P^*.

The algorithm described above computes the globally optimal partitioning for a sequence, but it cannot be computed until the complete sequence has been processed. For a practical implementation, we have to limit the maximum look-ahead in order to constrain the maximum processing delay.

Let s_{max} and s_{min} be the maximum and minimum number of frames to integrate into one sprite, respectively. Instead of constructing the complete computation graph for the entire sequence, we modify the algorithm such that a graph covering only the first s_{max} frames is generated. Moreover, all graph edges that span less than s_{min} frames are omitted. Note that this graph can be built online while new input

frames are received. When s_{max} frames are available, the minimum cost path is computed as before. This path again defines a partitioning, but now, we consider only the first segment. The subsequent processing stages can now begin to work in parallel on the sprite defined by the first segment. As new input images arrive at the multi-sprite algorithm, it again constructs the graph for the next s_{max} frames.

This algorithm does generally not give a globally optimal solution, but it limits the maximum memory requirement and processing delay that is introduced by the sprite generation to s_{max} frames.

Segmentation System Architecture

The proposed automatic segmentation system (Fig. 1) has been tuned for typical consumer video content, like sports or home surveillance. It is composed of several algorithms (see [3] for more detailed information):

- **Feature-based motion estimation.** This module computes interframe camera-motion parameters for pairs of successive frames.
- **Multi-sprite partitioning.** Given the set of interframe motion parameters, the multi-sprite partitioning separates the input sequence in segments for which independent background images should be synthesized.
- **Direct motion estimation.** Construction of a static background image requires global-motion parameters with high accuracy. This step refines the motion parameters with a gradient descent algorithm.
- **Background image reconstruction.** This step combines the input frames into a global background image and eliminates the foreground objects from this background image. A pure background image is obtained by temporal filtering of the camera-motion compensated input frames.
- **Background subtraction.** This step computes the video-object masks by comparing the input images with the pure background image. It also incorporates global-motion parameters to compensate the camera motion.

The processing in this framework is organized in a pipeline structure, where each stage operates concurrently on one segment of the input sequence.

The first stage in the pipeline computes the interframe motion parameters. The multi-sprite algorithm is interleaved to compute the next segment size whenever s_{max} frames are buffered. Once the segment length is known, the second stage can begin to compute accurate motion parameters for each input frame of the segment. The input images for the second stage are taken from the image buffers that delayed the input by s_{max} frames.

Stage 3 uses the accurate motion parameters to synthesize a background image. When Stage 3 finishes the synthetization of the background image, the image is moved to a queue of sprite-image buffers, such that Stage 3 can reconstruct the next sprite image while Stage 4 works on the segmentation of the previously computed segments.

If the system generates a sequence of short segments, several sprite-image buffers are required to store these images. To limit the number of buffers, we can limit the minimum length of a segment to include at least s_{min} frames. With this limitation, we only require s_{max}/s_{min} sprite-image buffers in the worst case.

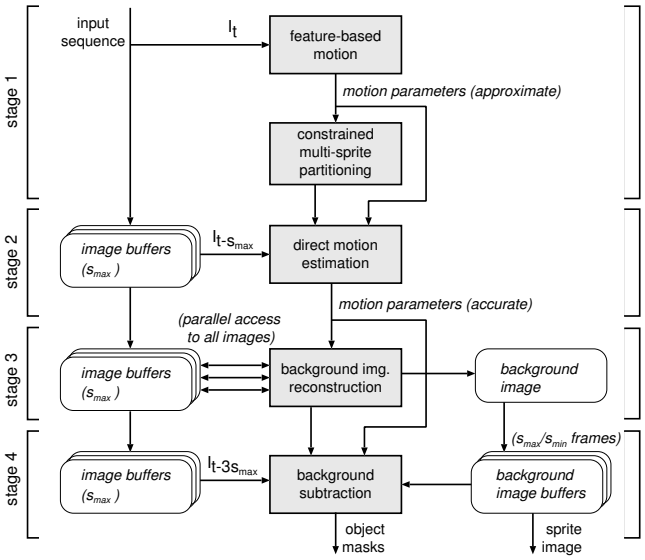

Fig. 1. The proposed online segmentation system. The maximum number of frames that are covered by a single sprite is limited to s_{max} frames. Hence, the processing can be pipelined with image-buffers (each with space for s_{max} images) at processing stages 2-4 and sprite-image buffers at stage 4.

Results and Conclusions

Experiments with various sequences (Fig. 2) show that a delay of $s_{max} \approx 50 - 100$ is sufficient to obtain a nearly optimal partitioning. Given the fact that video analysis can be carried out on monochrome images at CIF resolution, this results in a total memory usage of about $16 - 32$ MB. Hence, with this approach, it becomes feasible to integrate real-time video analysis with a processing delay of only 2-4 seconds into many consumer products, like video hard-disk recorders with integrated data-bases, or surveillance systems.[1]

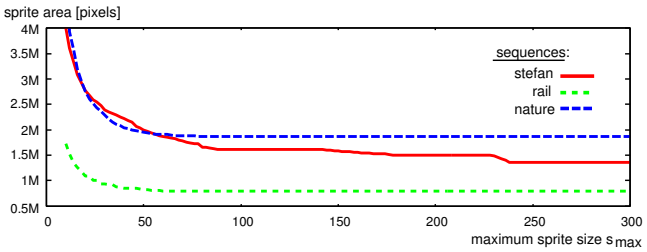

Fig. 2. Generated total sprite sizes for varying s_{max}.

References

[1] T. Aach and A. Kaup. Bayesian algorithms for adaptive change detection in image sequences using markov random fields. *Signal Processing: Image Communication*, 7:147–160, 1995.

[2] D. Farin, P. H. N. de With, and W. Effelsberg. Minimizing MPEG-4 sprite coding-cost using multi-sprites. In *SPIE Visual Communications and Image Processing (VCIP)*, volume 5308, pages 234–245, Jan. 2004.

[3] D. Farin, P. H. N. de With, and W. Effelsberg. Video-object segmentation using multi-sprite background subtraction. In *IEEE International Conference on Multimedia and Expo (ICME)*, volume 1, pages 343–346, June 2004.

[4] K. Jinzenji, H. Watanabe, S. Okada, and N. Kobayashi. MPEG-4 very low bit-rate video compression using sprite coding. In *Proc. IEEE International Conference on Multimedia and Expo*, page 2, Aug. 2001.

[1] Example results can be found at `http://vca.ele.tue.nl/demos/segmentation`.