Detection and Summarization of Salient Events in Coastal Environments*

D. Cullen, J. Konrad, and T.D.C. Little
Department of Electrical and Computer Engineering, Boston University
8 Saint Mary's St., Boston, MA 02215, USA
{*dcullen,jkonrad,tdcl*}*@bu.edu*

September 02, 2012

MCL Technical Report No. 09-02-2012

Abstract–The monitoring of coastal environments is of great interest to biologists and environmental protection organizations with video cameras being the dominant sensing modality. However, it is recognized that video analysis of maritime scenes is very challenging on account of background animation (water reflections, waves) and very large field of view. We propose a practical approach to the detection of three salient events, namely boats, motor vehicles and people appearing close to the shoreline, and their subsequent summarization. Our approach consists of three fundamental steps: region-of-interest (ROI) localization by means of behavior subtraction, ROI validation by means of feature-covariance-based object recognition, and event summarization by means of video condensation. The goal is to distill hours of video data down to a few short segments containing only salient events, thus allowing human operators to expeditiously study a coastal scene. We demonstrate the effectiveness of our approach on long videos taken at Great Point, Nantucket, Massachusetts.

Keywords: Video analytics, video summarization, ecological observation, remote cameras, webcam, Great Point Lighthouse, greatpointcam.org.

*In *Proc. 9th IEEE Intl. Conf. on Advanced Video and Signal-Based Surveillance,* September 2012. This work is supported by in part by MIT SeaGrant, "Consortium for Ocean Sensing of the Nearshore Environment." Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of MIT SeaGrant.

1 Introduction

Technological improvements of the last decade have made digital cameras ubiquitous. They have become physically smaller, more power-efficient, wirelessly-networked, and, very importantly, less expensive. For these reasons, cameras are finding increasing use in new surveillance applications, outside of the traditional public safety domain, such as in monitoring coastal environments [5]. At the same time, many organizations are interested in leveraging video surveillance to learn about the wildlife, land erosion, impact of humans on the environment, etc. For example, marine biologists would like to know if humans have come too close to seals on a beach and law enforcement officials would like to know how many people and cars have been on the beach, and if they have disturbed the fragile sand dunes.

Figure 1: Example of a detected boat and truck (top row) and a summary frame (bottom row) that shows both objects together.

However, with 100+ hours of video recorded by each camera per week, a search for salient events by human operators is not sustainable. Therefore, the goal of our research is to develop an integrated approach to analyze the video data and distill hours of video down to a few short segments of only the salient events. In particular, we are interested in identifying the presence of boats, motor vehicles and people in close proximity to the shoreline. This choice of objects of interest is dictated by our application but our approach is general and can be applied in other scenarios as well.

Although to date many algorithms have been proposed for moving object localization and recognition, very few approaches deal explicitly with the marine environment [12, 10]. Among the many approaches to video summarization we note key-frame extraction, montage and synopsis. While key-frame extraction [7], by design, loses the dynamics of the original video, video montage [3]

can result in loss of context as objects are displaced both in time *and* space. Video synopsis [8] does not suffer from the loss of context but is conceptually-complex and computationally-intensive.

There are three challenges to the detection and summarization of events in coastal videos. First, video analysis of maritime scenes poses difficulties because of water animation in the background, dune grasses blowing in the wind in the foreground, and the vast field of view. Secondly, a reliable recognition of objects from large distances (small scale) and especially of boats at sea is highly non-trivial. Thirdly, it is not obvious how to effectively summarize salient events for viewing by human operators.

We address these three challenges as follows. We rely on the movement of boats, motor vehicles and people for their detection. While difficult at small scales, motion detection is particularly challenging for boats at sea due to waves. We attack this problem by recognizing the fact that over longer time scales water movement can be considered stationary while boat movement is locally non-stationary both spatially and in time. In this context, we leverage the so-called *behavior subtraction* that has been shown to effectively deal with animated water while accurately detecting boat movement [9]. The obtained moving-object masks serve as regions of interest (ROIs) for subsequent boat, motor vehicle and people detection. We apply the *feature-covariance framework* [11] to reliably distinguish between different objects even at small scales. Having detected and classified salient events, the last step is their summarization. We adapt the so-called *video condensation* [4] to our needs by devising a suitable cost needed by the algorithm based on the outcome of behavior subtraction and object detection.

Our paper focuses on a coastal environment at Great Point, Nantucket, MA. We have recorded hundreds of hours of video from networked cameras located close to the beach (Figure 1). Below, we demonstrate the effectiveness of our algorithms on this dataset with condensation ratios approaching 20:1, thus dramatically reducing the need for human operator involvement in search for events of interest.

2 Proposed Approach

The goal of our system is to automatically distill hours of video down to a few short segments containing only the types of objects sought. Our system must be simple enough to be used by operators with no expertise in image processing or video surveillance. Furthermore, we want a system that can be easily extended to detect new types of objects. The final summarization must be natural and useful to biologists, environmental protection agents, etc.

Our system, whose block diagram is shown in Figure 2, meets these requirements. Each of the processing blocks requires only a few parameters that are easy to tune; a fairly wide range of values will produce reasonable results. To extend the system to detect new types of objects, the user needs only to provide new images similar to the target class of objects of interest. Finally, any combination of classes of objects can be selected for summarization, which gives the operator a lot of control over the output.

We assume that video is acquired by a non-PTZ camera, no camera vibration takes place, and there is no moisture on the lens (from rain or fog). While global motion compensation can help with the first two issues, the third one is more challenging and needs to be addressed in the future.

Our system first runs background subtraction to detect the ROI (moving) areas in the video, and follows with behavior subtraction to reduce the amount of uninteresting activity such as ocean

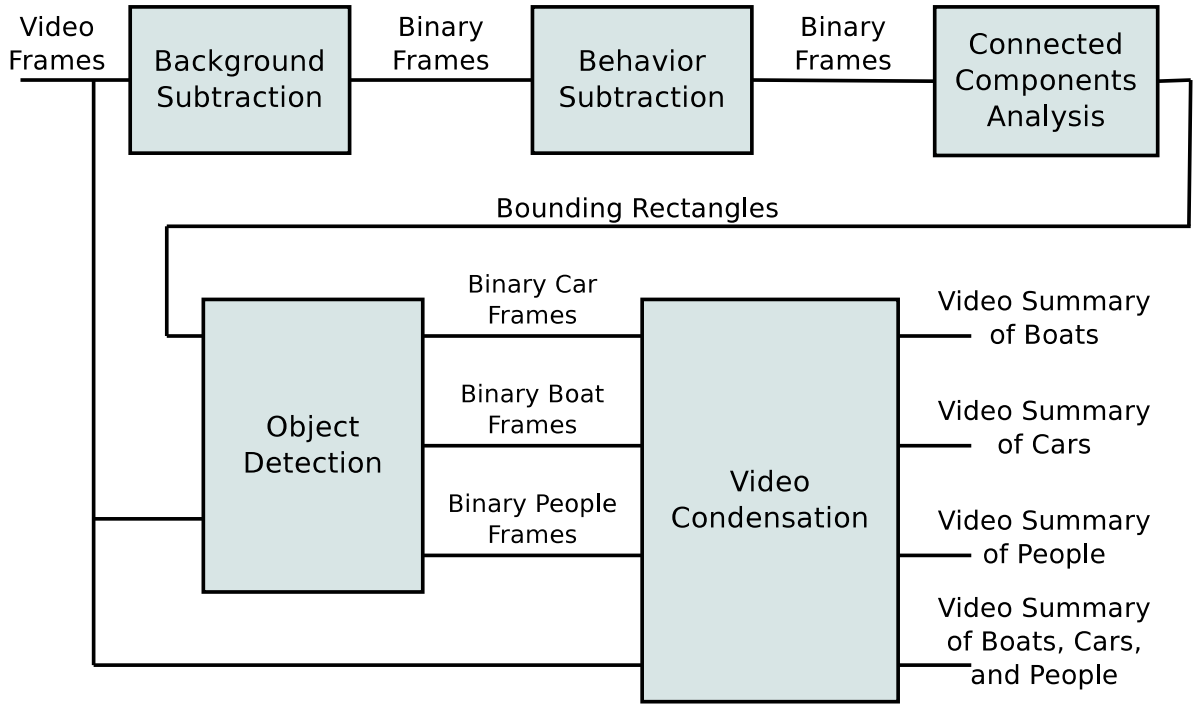

Figure 2: Block diagram of the entire system.

waves. Next, the system runs a connected-component analysis on the behavior-subtracted (binary) video to label the regions where moving objects are likely. Then, the corresponding regions in video frames are tested for the presence of objects of interest using a covariance-based classifier. The regions classified as either boats, cars or people are then used as high-cost areas that cannot be removed in the subsequent video condensation.

The following sections describe each step of the overall system. However, for the sake of brevity, many details have been omitted, so we refer the reader to the literature.

2.1 Background Subtraction

We detect moving objects in the camera field of view by means of background subtraction. Many background subtraction algorithms have been developed to date [2] with more advanced ones requiring significant computational resources. Since our system needs to process many hours of video, it is critical that background subtraction be as efficient as possible. Therefore, we use a very simple and computationally-efficient background model in combination with a Markov model for the detected labels L [6]. First, an exponentially-smoothed moving average is used to estimate background B^k by means of recursive filtering:

$$B^k[\boldsymbol{x}] = (1-\alpha) * B^{k-1}[\boldsymbol{x}] + \alpha * I^k[\boldsymbol{x}] \tag{1}$$

where I^k is the input video frame, k is the frame number, $\boldsymbol{x} = [x\ y]^T$ are pixel coordinates, and α is a smoothing parameter. The computed background B^k is then used in the following hypothesis test on the current video frame:

$$|I^k[\boldsymbol{x}] - B^{k-1}[\boldsymbol{x}]| \underset{\mathcal{S}}{\overset{\mathcal{M}}{\gtrless}} \theta \exp\left((Q_{\mathcal{S}}[\boldsymbol{x}] - Q_{\mathcal{M}}[\boldsymbol{x}])/\gamma\right) \tag{2}$$

where $Q_{\mathcal{S}}$ and $Q_{\mathcal{M}}$ denote the number of static ($\mathcal{S}$) and moving ($\mathcal{M}$) neighbors of $\boldsymbol{x}$, respectively, θ is a threshold and γ is a tuning parameter that adjusts the impact of Markov model. Note that the overall threshold in the above hypothesis test depends on the status of neighboring pixels. If there are more moving than static neighbors, then the overall threshold is reduced thus encouraging an $\mathcal{M}$ label at $\boldsymbol{x}$. The above test produces moving/static labels L^k for each video frame I^k. In our experiments, we have used 2nd order Markov neighborhood (8 nearest pixels). The second row in Figure 4 shows typical results for the above algorithm.

2.2 Behavior Subtraction

In our coastline videos, there is a great deal of repetitive background motion, such as ocean waves and dune grass, that results in spurious detections after background subtraction. However, this motion can be considered stationary (in a stochastic sense), when observed over a longer period of time, and suitably characterized. One method to accomplish this is the so-called behavior subtraction [9] that we leverage here due to its simplicity and computational efficiency. The algorithm operates on binary frames of labels L produced by background subtraction (Figure 2) and has two phases: training and testing. In the training phase, a segment of N frames from an M-frame training sequence ($M \geq N$) is being examined. The training sequence is assumed to exhibit the stationary dynamics we want to remove, but no "interesting" moving objects. First, labels L at each pixel are accumulated across all frames of the N-length segment to form a 2D array. Then, all such 2D arrays (for all N-frame segments within M-frame training sequence) are compared to select the maximum at each pixel. The resulting 2D training array constitutes the description of stationary scene dynamics. In the testing step, a sliding N-frame segment is used to accumulate labels L, from the background-subtracted sequence being processed, in a new 2D test array. If a pixel in the test array exceeds the value of the same pixel in the training array (maximum across many segments) by a threshold of Θ, then a detection is declared (white pixel). The rationale behind this is that an "interesting" object will occupy those pixels for more frames than the maximum number of frames occupied by stationary behavior, thereby allowing us to remove regions with "uninteresting" motion, such as waves. Typical results of behavior subtraction are shown in the second row of Figure 4.

2.3 Region of Interest (ROI) Extraction

We use a simple connected-components algorithm to label regions in the behavior-subtracted video. Each such region potentially includes a moving object. Since the classification algorithm (next section) operates on rectangular regions of pixels for efficiency, we circumscribe an axis-aligned bounding box around each of the connected components. We discard any bounding boxes smaller than a given threshold (5×5 in our experiments) as too small to contain any of the target objects. We also increase the size of each bounding box by a constant scale factor (20% in our experiments) to ensure that it completely captures the target object. This margin is important because the connected components may be misaligned with the underlying video objects and we risk losing part of the object causing subsequent mis-detection. The resulting bounding boxes are passed to the object detection step.

2.4 Object Detection Using Feature Covariance

The bounding boxes identify video frame areas that likely contain moving objects of interest. The goal now is to verify whether each box contains either a boat, a motor vehicle or a person, or, alternatively, is a false alarm, e.g., due to an ocean wave. We employ a method developed by Tuzel *et al.* [11] that compares covariance matrices of features. This approach entails computing a d-dimensional feature vector for every pixel in a region, and then generating a $d \times d$ covariance matrix from all such feature vectors. We use rectangular regions of pixels for computational efficiency but any shape region can be used. The degree of similarity between two regions is computed by applying a distance metric to their respective covariance matrices.

In order to detect objects in an image, Tuzel *et al.* apply exhaustive search where each object from a dictionary is compared with query rectangles of different sizes at *all* locations in the searched image. If the dictionary object and the query rectangle are sufficiently similar, the object is detected. However, even with the use of integral images [11], this approach is computationally expensive, especially when each video frame must be searched for many different sizes, locations and classes of objects. Instead, we use ROIs identified by the bounding boxes at the output of connected-component analysis (Figure 2) as the query rectangles and test each against three dictionaries: boats, motor vehicles, and people (described below). This greatly reduces the number of queries, thus increasing the search speed. Another advantage of our approach is the automatic selection of rectangle size (from the connected-component analysis). In contrast, Tuzel *et al.* use several fixed-size query rectangles, thus making an implicit assumption about the size of target objects. In our experiments, we used

$$\vec{\xi}(\boldsymbol{x}) = \left[x, y, \left|\frac{\partial I[\boldsymbol{x}]}{\partial x}\right|, \left|\frac{\partial I[\boldsymbol{x}]}{\partial y}\right|, \left|\frac{\partial I^2[\boldsymbol{x}]}{\partial x^2}\right|, \left|\frac{\partial I^2[\boldsymbol{x}]}{\partial y^2}\right|\right] \tag{3}$$

as the feature vector. It contains location, and first- and second-order derivatives of intensity I, but is void of color attributes since we believe shape of objects of interest is more informative than their color.

Similarly to Tuzel *et al.*, we use the nearest-neighbor classifier to detect the objects. For each bounding box detected in a video frame (ROI), we compute the distances between its covariance matrix and the covariance matrix of each object in the three dictionaries. If the minimum distance is below a threshold ψ, then the class of the object from the dictionary corresponding to this minimum distance is assigned to the ROI. We repeat this procedure for all the detected bounding boxes.

Since covariance matrices do not lie in a Euclidean space, as the distance metric between two covariance matrices we use the Frobenius norm of the difference between covariance matrix logarithms proposed by Arsigny *et al.* [1], that is often referred to as the log-Euclidean metric.

For each class of objects we wish to detect, we created a dictionary composed of many representative images (samples are shown in Figure 3). We downloaded images of assorted sizes from Google Images and cropped them to remove extraneous background clutter. We tried to diversify the selection to assure that our dictionaries are representative of real-life scenarios. For example, our motor vehicles dictionary consists of SUVs, pick-up trucks, sedans, etc., at various orientations. The dictionaries we have used in our tests are composed of 30-100 images.

(a) (b) (c)

Figure 3: Two samples from each of the dictionaries: (a) boats, (b) motor vehicles, and (c) people used in feature-covariance detection. Images have been obtained from a search on Google Images.

2.5 Video Condensation

After each ROI has been identified as either a boat, motor vehicle or person, or, alternatively, ignored, a compact summary of all objects moving in front of the camera needs to be produced. To this effect, we employ the so-called video condensation [4], a method that is capable of shortening a long video with sparse activity to a short digest that compactly represents *all* of the activity while removing inactive space-time regions. Since the method is quite involved, we refer the reader to the original paper [4] and describe here only the main concepts.

Video condensation takes a video sequence and an associated cost sequence as inputs. The cost sequence is used to decide whether to remove specific pixels from the video sequence or not. Although any cost can be used, we are interested in preserving the detected boats, cars or people while removing areas void of activity and thus we use the originally-proposed cost [4], namely moving/static labels at the output of behavior subtraction. More specifically, we use only those labels that are within bounding boxes of the detected boats, cars or people; the pixels outside are assigned 0 (no activity) and thus can be removed.

Consider a boat at sea traveling to the right in the field of view of a camera whose scan lines are aligned with the horizon (Figure 4). After the behavior subtraction step we obtain, ideally, a silhouette of the moving boat in each video frame. Jointly, all these silhouettes form a silhouette tunnel that is at some angle to the time axis t in the $x - y - t$ space of the video sequence (video cube). This tunnel starts at the left edge of the video cube and ends at the right edge (converse is true for a boat moving to the left). If another boat follows after the first boat disappears, another tunnel will be formed with a "dead" space, where is no activity, between them. In the first stage, video condensation eliminates this "dead" space by removing all frames void of silhouettes. In consequence, as soon as one tunnel finishes (boat exits on right), another tunnel starts (boat enters on left). In the next stage, ribbons rather than frames are removed. Ribbons are connected surfaces that, for the above case of horizontal motion, are rigid vertically and can flex horizontally between consecutive time instants (see [4] for details). This ability to flex in the $x - t$ plane (while remaining rigid vertically) allows ribbons to "fit" between two silhouette tunnels. If a ribbon cuts through

"dead" space of no activity and is removed, the two tunnels get closer to each other. After recursively repeating this removal, the two tunnels almost touch each other which is manifested in the condensed video by the two corresponding boats appearing in the same video frame despite being far apart in time in the original video. Ribbons with increasing degree of flexibility can be used to remove topologically more complex "dead" space. Here, we first use rigid ribbons, or frames (flex-0), and follow with progressively more and more flexible ribbons (flex-1, flex-2, and flex-3) [4].

2.6 Implementation Details

All code was written in C++ for computational efficiency and portability to embedded sensing platforms in the future. We used the open-source FFMPEG libraries for video decoding and encoding, and the simple OpenCV library function *cvFindContours* for connected-component analysis. All other code was written "from scratch".

3 Experimental Results

We have collected hundreds of hours of coastal videos at 640×360 resolution and 5fps from cameras mounted at Great Point, Nantucket Island, MA. Figure 4 shows two examples from one of the videos we processed. We used the following ranges for parameters: background subtraction – $\alpha = 0.005$, $\theta = 20 - 25$, $\gamma = 1.0$, 2nd-order Markov model with 2 iterations; behavior subtraction – $M = 300 - 2500$, $N = 50 - 100$, $\Theta = 0.0 - 1.0$; connected-component analysis – 5×5 bounding box threshold, 20% box enlargement; object detection – dictionaries for boats, cars and people consisting of 30, 80 and 62 objects, respectively, with corresponding thresholds ψ between 2.5 and 4.0. The large range for M in behavior subtraction (training set size) is needed to adapt to weather conditions (see below).

Note the relative resilience of background subtraction to the presence of waves (second row in Figure 4). Although behavior subtraction provides only a slight additional wave suppression in this case, for videos with choppy water surface behavior subtraction offers a significant improvement for larger values of M. Also, note the detection of the boat on left despite a long wake behind it. The condensed frame on left (bottom row) shows two boats together that were never visible in the field of view of the camera at the same time. Similarly, on right, four boats have been combined into one condensed frame thus shortening the overall video. We invite the reader to view the original and condensed videos on our project's page.

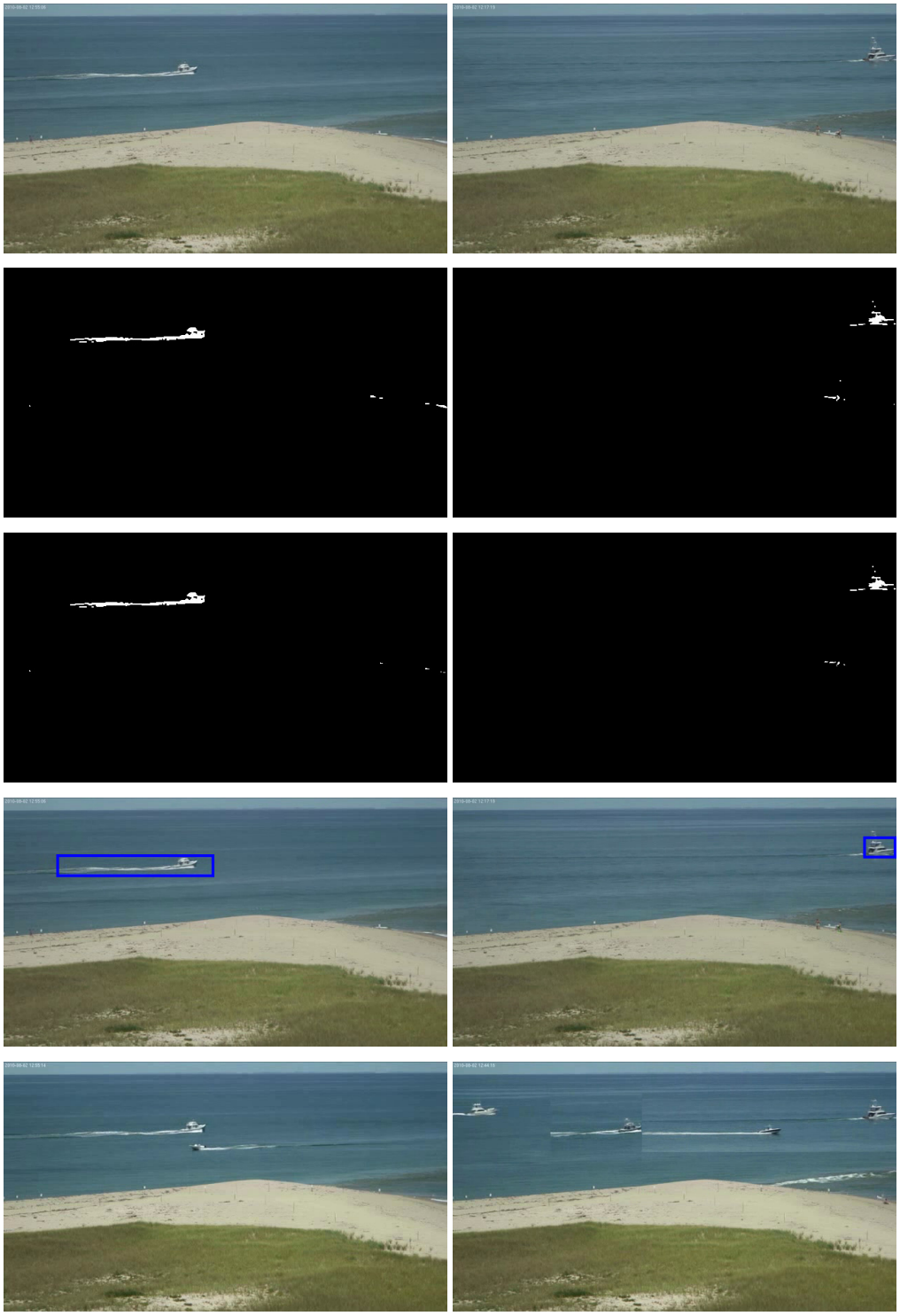

Figure 4: Samples of typical input video frames (top row) and outputs from the processing blocks in Figure 2: background subtraction (row 2), behavior subtraction (row 3), object detection (row 4) and video condensation (row 5).

The effectiveness of our joint detection and summarization system can be measured by the condensation ratio (CR) achieved for each class of objects (or combination thereof). Table 1 shows detailed results with cumulative condensation ratios (after flex-3) of over 18:1 for boats, about 9:1 for people, but only 6:1 for boats or people. Clearly, when one wants to capture a larger selection of objects, the condensation ratio suffers. Condensation ratios for another video with boats, cars and people are shown in Table 2. The low CR for cars (3:1) is due to the fact that an SUV and pick-up truck arrive and remain on the beach until the end of the video. The very high CR for people (61:1) is due to the fact that, although passengers exit the vehicles, they remain next to them and are often not detected. Note the last row in both tables labeled "beh. subtr." with very low CRs. These are results for the whole-frame behavior subtraction output being used as the cost in video condensation instead of being limited to the bounding boxes of detected objects. Clearly, the spurious detections at the output of behavior subtraction reduce the condensation efficiency.

Table 1: Number of frames after each flex-step and cumulative condensation ratios (CR) for 38-minute, 5fps video with boats and people (11,379 frames after behavior subtraction).

Cost	# of frames after each step				CR
	flex-0	flex-1	flex-2	flex-3	
boats (B)	1752	928	723	600	18.97:1
people (P)	3461	2368	1746	1285	8.85:1
B or P	4908	3253	2504	1897	5.99:1
beh. subtr.	11001	8609	8147	7734	1.47:1

Table 3 provides the average execution time for each stage of processing in a single-threaded C++ implementation on an Intel Core i5 CPU with 4GB of RAM running Ubuntu 11.04 Linux. Note that the execution times for background subtraction and behavior subtraction depend only on the video resolution (in this case, 640×360). On the contrary, the execution times of object detection and video condensation vary depending upon the data (e.g., more activity means that more regions must be checked for the presence of objects and also that fewer frames can be dropped in the flex-0 stage of condensation). The benchmarks reported in the table were obtained for detections of cars in the video from Table 2 and are representative of typical execution times. Since video condensation operates on blocks of frames, we computed the average processing time of each "flex" pass by dividing the execution time for that pass by the number of input frames to that pass. For the background subtraction, we used second-order Markov neighborhood and two update iterations for each frame. Disabling the MRF model reduces the execution time to 0.156 sec/frame but significantly lowers the quality of the output. The object detection benchmark in the table includes the time for the connected components computation, the bounding box extraction, and the tests of each candidate region against three dictionaries (cars, people, and boats).

Clearly, our single-threaded implementation can process only 0.2fps. Even for a 5fps input video this is far from real time. One possibility to close this gap is by leveraging parallelism. For example, we have experimented with a pipelined, multithreaded implementation of video condensation for different flex parameters. We found that on a quad-core CPU this reduced the processing time by up to a factor of three for typical hour-long beach videos, compared to a traditional single-core approach. Similar parallelism can be applied to the other steps.

Table 2: Number of frames after each flex-step and cumulative condensation ratios (CR) for 22-minute, 5fps video with boats, cars and people (6,500 frames after behavior subtraction).

Cost	# of frames after each step				CR
	flex-0	flex-1	flex-2	flex-3	
cars (C)	2273	2173	2137	2070	3.14:1
boats (B)	633	465	452	435	14.94:1
people (P)	158	117	114	107	60.75:1
C or B or P	2865	2667	2614	2561	2.53:1
beh. subtr.	6437	5843	5619	5460	1.19:1

Table 3: Average execution time for each stage of processing.

Processing Step	Average Execution Time
Background Subtraction	0.292 sec/frame
Behavior Subtraction	0.068 sec/frame
Object Detection	0.258 sec/frame
Video Condensation	
flex 0	0.034 sec/frame
flex 1	2.183 sec/frame
flex 2	1.229 sec/frame
flex 3	0.994 sec/frame
Total	5.058 sec/frame

4 Summary and Conclusions

In this paper, we combined multiple video processing algorithms to create a flexible, robust coastal surveillance system that should be useful for marine biologists, environmental agents, etc. We tested our approach extensively using real coastal video sequences and showed that our system can reduce the length of typical videos up to about 20 times without losing any salient events. Our system can dramatically reduce human operator involvement in search for events of interest. Currently, our system does not operate in real time but with a careful implementation on a multicore architecture real-time performance is within reach.

In the course of our research, we have made a few observations. Although a fixed dictionary for each class of objects extracted from Google Images has worked quite well, we managed to obtain better results by manually creating each dictionary from videos in our dataset (different from the video under analysis but captured by the same camera). This is due to the similarity of viewpoint, distance, background between all videos. This should be useful in practice but would require operator involvement to populate/prune the dictionaries. We also observed that the detection step works better if objects with significantly different shape are treated as a separate class. For example, the sailboats have large triangular shape, which is quite different from the hull/cabin of motorboats; the detection works best in this case if we use a separate dictionary for each type of boat. Finally, a more advanced classification method than the nearest-neighbor (NN) classifier tested here can be applied. For example, in very preliminary tests a kNN classifier showed a marked improvement over the NN classifier.

References

[1] V. Arsigny, P. Pennec, and X. Ayache. Log-Euclidean metrics for fast and simple calculus on diffusion tensors. *Magnetic Resonance in Medicine*, 56(2):411–421, 2006.

[2] N. Goyette, P.-M. Jodoin, F. Porikli, J. Konrad, and P. Ishwar. changedetection.net: A new change detection benchmark dataset. In *IEEE Workshop on Change Detection (CDW'12) at CVPR'12*, June 2012.

[3] H.-W. Kang, Y. Matsuhita, X. Tang, and X.-Q. Chen. Space-time video montage. In *Proc. IEEE Conf. Computer Vision Pattern Recognition*, pages 1331–1338, June 2006.

[4] Z. Li, P. Ishwar, and J. Konrad. Video condensation by ribbon carving. *IEEE Trans. Image Process.*, 18(11):2572–2583, Nov. 2009.

[5] T. Little, P. Ishwar, and J. Konrad. A wireless video sensor network for autonomous coastal sensing. In *Proc. Conf. on Coastal Environmental Sensing Networks (CESN)*, Apr. 2007.

[6] J. McHugh, J. Konrad, V. Saligrama, and P.-M. Jodoin. Foreground-adaptive background subtraction. *IEEE Signal Process. Lett.*, 16(5):390–393, May 2009.

[7] J. Oh, Q. Wen, J. Lee, and S. Hwang. Video abstraction. In S. Deb, editor, *Video Data Mangement and Information Retrieval*, chapter 3, pages 321–346. Idea Group Inc. and IRM Press, 2004.

[8] Y. Pritch, A. Rav-Acha, and S. Peleg. Non-chronological video synopsis and indexing. *IEEE Trans. Pattern Anal. Machine Intell.*, 30(11):1971–1984, Nov. 2008.

[9] V. Saligrama, J. Konrad, and P.-M. Jodoin. Video anomaly identification: A statistical approach. *IEEE Signal Process. Mag.*, 27(5):18–33, Sept. 2010.

[10] A. Samama. Innovative video analytics for maritime surveillance. In *Int. Waterside Security Conference*, Nov. 2010.

[11] O. Tuzel, F. Porikli, and P. Meer. Region covariance: A fast descriptor for detection and classification. In *Proc. European Conf. Computer Vision*, May 2006.

[12] Q. Wu, H. Cui, X. Du, M. Wang, and T. Jin. Real-time moving maritime objects segmentation and tracking for video communication. In *Int. Conf. on Communcation Technology*, Nov. 2006.