Achieving Real-Time Object Detection and Tracking Under Extreme Conditions

Fatih Porikli

Abstract—**In this survey, we present a brief analysis of single camera object detection and tracking methods. We also give a comparison of their computational complexities. These methods are designed to perform accurately under difficult conditions such as erratic motion, drastic illumination change, and noise contamination.**

I. Introduction

OBJECT tracking is one the most important tasks in computer vision. In video surveillance, it assists understanding the movement patterns of people to uncover suspicious events. It is a key technology in traffic management to estimate flux and congestion statistics. Advanced vehicle control systems depend on the tracking information to keep the vehicle in lane and prevent from collisions. In physical therapy, analyzing the mobility of patients improves the accuracy of their diagnosis. Learning the shopping behavior of customers by tracking assists the architecture design in retail space instrumentation. In robotics, tracking bridges the gap between the raw visual information and environmental awareness. In video summarization, it is applied to generate object-based representations and automatic content annotations. Tracking is also a fundamental technology to extract regions of interest and video object layers as defined in JPEG-2000 and MPEG-4 standards.

Even though it is essential to many applications, robust object tracking under uncontrolled conditions still poses a challenge. Real-life systems are required to track objects not only when the background scene is static but also when lighting changes suddenly, camera-object motion becomes large, color contrast becomes low, image noise soars to an unacceptable level, etc. In addition, the computational complexity is required to be kept minimum for real-time performance.

In the following sections, we describe object detection and tracking methods that are designed to resolve the above issues.

II. Object Detection

Background subtraction is a common approach for discriminating moving regions in fixed camera setups. Basically, a pixel-wise reference model for the stationary part of the scene is estimated. Then, the observed image is compared with this reference to obtain a foreground mask.

A reference frame can be easily computed by averaging (or α-blending) the previous frames. However, averaging induces ghost effect, i.e. distortion of colors behind moving objects. Illumination changes, moving shadows, and other motions e.g. swaying trees also cause similar artifacts. Alternatively, predictive techniques such as Kalman [?] and Wiener [?] filters are applied to learn the temporal color wariations at each pixel. These techniques assume color variations can be modeled by a random variable. They heavily depend on the predefined state transition parameters. As a result, they may fail in case the color distribution does not fit into a single model or color distribution has a different shape function.

To handle multimodal backgrounds, mixture of model background is proposed [?]. Often, these models are assigned as Gaussian functions. Since the background models is updated at every frame, iterative update mechanisms, e.g. online expectation maximization (EM) algorithm, are applied to fit the models. However, online EM blends weak modes into stronger ones and distorts the model mean values as shown in Fig. 1. To achieve accurate adaptation of models, a Bayesian update mechanism [?], which can also estimate the number of required models, is proposed. This approach is flexible enough to handle illumination variations and other arbitrary changes in the scene. There are also variants of the mixture of model background that uses image gradient and optical flow information [?]. Mixture of model approaches can converge to any arbitrary distribution provided enough number of observations. However, their computational cost grows exponentially as the number of models in the mixture increases. Another background modeling approach is nonparametric kernel density estimation [?]. This method stores color values of multiple frames and estimates the contributions of a set of kernel functions using all of the data instead of iteratively updating background models at each frame. Both memory and computational cost are proportional to the number of frames. As a result, kernel based methods may become infeasible for real-time applications.

A shortcoming of the above methods is that they neglect the temporal correlation of color values. This impedes distinguishing a periodic background motion such as swaying plants driven by wind, waves on a beach, rotating objects, etc., from the foreground object motion. Since real-world physics often induces near-periodic phenomenon in the environment, a frequency decomposition based representation of the background, *wave-back*, is proposed [?]. This algorithm detects motion based on the form of the temporal color variation by comparing the frequency transform responses. Thus, the background models are fit in the frequency domain. Similar to kernel based methods, a multitude of frames are stored. At every frame, a frequency transform is applied and the obtained coefficients are compared to the background models to generate a distance map as given in Fig. 2. Detection of the time-varying phenomenon is also attempted using corner-based background models [?]. Instead of processing every pixel, this algorithm detects a certain number of feature points using

TABLE I

PERFORMANCE OF BACKGROUND BASED OBJECT DETECTION

	complexity	msec	LA[†]	PB[†]	MM[†]	N
α-Blend	$O(M)$	5	poor	poor	poor	p
Kalman [?]	$O(M)$	8	some	poor	poor	s
EM [?]	$O(Mn^2)$	45	some	poor	good	g
Bayesian [?]	$O(Mn^2)$	35	some	poor	best	g
Wave-back [?]	$O(MTlog(T))$	55	good	best	poor	p
Corner [?]	$O(MD^2)$	150	some	good	poor	g
Intrinsic [?]	$O(MT)$	30	best	good	poor	

([†]) LA: lighting adaptation, PB: periodic backgrounds, MM: multi-modal backgrounds, NH: noise handling. See text for parameters.

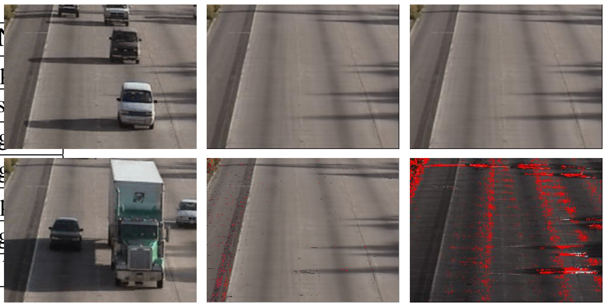

Fig. 1. EM update [?] vs. Bayesian update [?]. **Left:** Sample frames. **Middle:** First (top) and second (bottom) background layers of EM update. **Right:** Bayesian update result. EM update inaccurately blends distinct modes into identical layers. Bayesian update can identify separate layers, i.e. road and shadow modes. (Red means no layer)

a Harris corner detector and a scale-invariant feature point descriptor. It dynamically learns a single background model and classify each extracted feature as either a background or a foreground feature. It also uses a "Lucas-Kanade" motion tracker to differentiate motion consistent foreground points from background points with random or repetitive motion. Since feature point extraction and descriptor generation are computationally expensive, this algorithm may not be suitable for real-time applications if number of feature points increases. Sample detection results are shown in Fig. 3.

Instead of adapting models to the background, the scene can be represented using intrinsic images as a multiplication of static and dynamic parts [?]. This algorithm keeps a set of previous images and applies gradient filters. By taking advantage of the sparseness of the filter responses, it estimates the background intrinsic image using a log domain median filter. From background and current image, it constructs a foreground mask as shown in Fig. 4. This method is robust to sudden and severe illumination changes and computationally inexpensive. Sample comparison results with mixture models are provided in Fig. 5.

We summarized the performance of the above algorithms under real-life conditions as well as their computational complexity in table I. We denote the image size (total number of pixels) as M, the number of background models as n, the number of previous frames as T, and the local descriptor window size as D (other constants are disregarded). We also list the speed of each algorithm running on a P4 3GHz machine for a 320×240 color video. Note that, the computational load and memory requirements are heavily coupled and they depend on whether a hardware implementation (or parallel processing) is adapted or not. Usually, the following techniques are employed to reduce the computational complexity and scale down the memory footprint for practical systems:

- Partially updating the background or subsampling,
- Limiting temporal size of detectable variations,
- Assuming color channels to be independent,
- Using lower dimensional color representations.

III. INTER-FRAME TRACKING

Tracking, that is finding a region corresponding to a given object in the image, also faces similar challenges. Objects frequently change their appearance and pose. They occlude

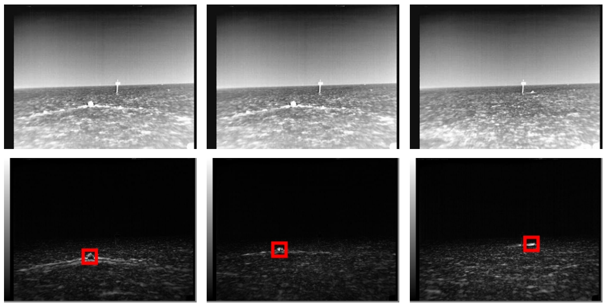

Fig. 2. Object detection in thermal IR that depicts a sea shore. As visible, the wave-back [?] method distinguishes the periodic motion of sea waves from the motion of a small boat. *(Courtesy of PETS)*

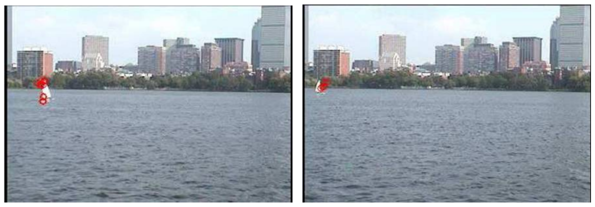

Fig. 3. Detected foreground points by corner based approach [?].

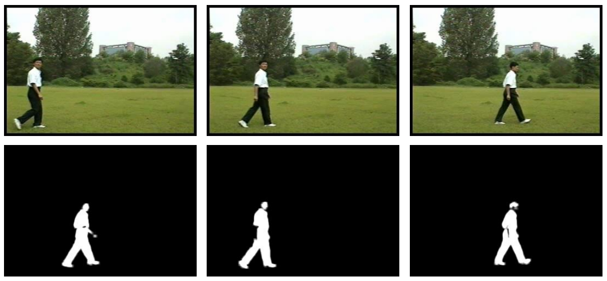

Fig. 4. Detected foreground regions using intrinsic background [?].

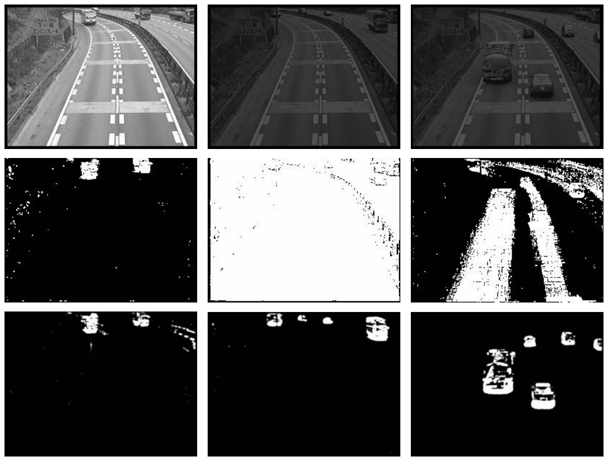

Fig. 5. Comparison of GMM [?] and intrinsic [?] background modeling. **Top:** Sudden illumination change happens. **Middle:** GMM method confuses and its recovery takes time. **Bottom:** Intrinsic background is not disturbed. Both methods use the RGB color space.

each other, become temporarily hidden, merge together and split. Depending on the application, they exhibit erratic motion patterns and often make sudden turns.

Tracking can be considered as estimation of the state given all the measurements up to that moment, or equivalently constructing the probability density function of object location. A simple tracking approach is predictive filtering. This method uses object color and location statistics while updating an object model by constant weights [?]. When the measurement noise are assumed to be Gaussian, the optimal solution is provided by the Kalman filter [?]. When the state space is discrete and consists of a finite number of states, Markovian filters can be applied for tracking. The most general class of filters is represented by particle filters, which are based on Monte Carlo integration methods. The current density of the state (which can be location, size, speed, boundary [?], etc.) is represented by a set of random samples with associated weights and the new density is computed based on these samples and weights. Particle filtering is a popular tracking method [?],[?]. However, it is based on random sampling that becomes a problematic issue due to sample degeneracy, especially for higher dimensional representations.

In contrast, the mean-shift tracker is a non-parametric density gradient estimator that is iteratively executed within the local search kernels [?]. It models the object probability density in terms of color histogram, and moves the object region towards the largest gradient direction. Thus, it is computationally simple. Nevertheless, if the object relocation between successive frames is larger than the kernel size, it fails to detect the object. Since the histograms are used to determine likelihood, the gradient estimation and convergence becomes inaccurate in case object and background color distribution are similar. To solve this issue, a multi-kernel mean-shift approach is proposed [?]. The additional kernels are obtained by background subtraction. In order to resolve the above convergence issue, another kernel that pushes the object away from the background regions are adapted.

Tracking can also be considered as a classification problem and a classifier can be trained to distinguish the object from the background [?]. This is done by constructing a feature vector for every pixel in the reference image and training a classifier to separate pixels that belong to the object from pixels that belong to the background. Integrating classifiers over time improves the stability of the tracker in cases illumination changes. As in the mean-shift, an object can be tracked only if its motion is small. This method can confuse objects in case of an occlusion.

Object representation, that is how to convert color, motion, shape, and other properties into a compact and identifiable form, plays critical role in tracking. Conventional trackers either depend only on color histograms, which disregard the structural arrangement of pixels, or appearance models, which ignore the statistical properties. There are several shortcomings of these representations. Populating higher dimensional histograms by a small number of pixels results in an incomplete representation. Besides, histograms are easily distorted by noise. Appearance models are sensitive to the scale changes and localization errors.

Covariance matrix representation [?] embodies both spatial and statistical properties of objects, and provides an elegant solution to fusion of multiple features. Covariance is an essential measure of how much the deviation of two or more variables or processes match. In tracking, these variables correspond to point features such as coordinate, color, gradient, orientation, and filter responses. This representation has much lower dimensionality than histograms. It is robust against noise and lighting changes as shown in Figs. 6-7. To track objects using covariance descriptor, an eigenvector based distance metric is adapted to compare the matrices of object and candidate regions [?]. Covariance tracker does not make any assumption on the motion. This means that it can keep track of objects even if their motion is erratic and fast. It can compare any regions without being restricted to a constant window size. In spite of these advantages, the computation of the covariance matrix distance for all candidate regions is slow and requires exponential time. An integral image based algorithm that requires constant time is proposed to improve the speed [?]. This technique significantly accelerates the covariance matrix extraction process by taking advantage of the spatial arrangement of the points as illustrated in Fig. 8. The graphs show the ratio of computational savings for histogram extraction, which in return accelerates the tracking methods that use histograms, and covariance matrix extraction.

Table II shows the computational load and robustness of the above tracking methods under various conditions including erratic motion, appearance changes, etc. We denote the tracked object size (number of pixels in object region) as N, the image size (total number of pixels) as M, the histogram size as H, the candidate regions as R, the number of features in the covariance matrix as F, and the number of classifiers as C. We also present the approximate processing times for tracking of a 20×40 object on a P4 3GHz machine.

As many vision tasks, object detection and tracking also benefit from specific hardware implementations. Such implementations contain various combinations of different subsystems such as traditional Digital Signal Processors (DSP),

TABLE II
PERFORMANCE OF INTER-FRAME TRACKING

	complexity	msec	AC†	EM†	FM†	SO†
Predictive [?]	$O(NH)$	10	some	poor	poor	poor
Mean-shift [?]	$O(NH)$	12	some	some	poor	poor
Multi-kernel [?]	$O(NHR)$	20	some	good	good	poor
Particle [?]	$O(NHR)$	25	good	some	poor	poor
Ensemble [?]	$O(NHC)$	20	best	some	poor	poor
Covariance [?]	$O(MF^2)$	150	good	good	best	some

AC: appearance changes, EM: erratic motion, FM: fast motion, SO: small objects. See text for parameters and explanation.

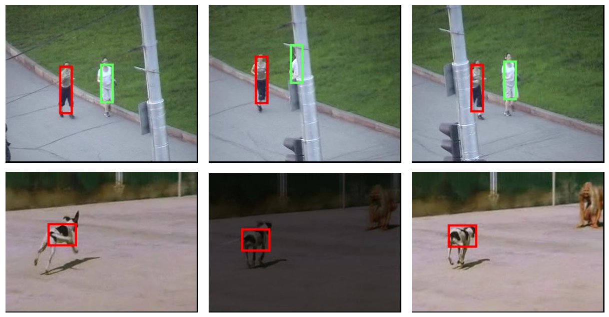

Fig. 6. Sample covariance tracking [?] results. Occlusion (top) and severe illumination change (bottom). Note that, intensity is not discarded as in conventional approaches, in contrast, the RGB color space is used.

Graphic Processor Units (GPU), Fully Programmable Gate Arrays (FPGA), Application Specific Integrated Circuits (ASIC), and other reconfigurable cores.

DSPs offers software programmability, which is a cost-effective means for keeping hardware viable. With a programmable DSP architecture, it is possible to speed up fundamental low-level algorithms. On the other hand, ASICs offer a high performance, low power, and low cost option for implementing algorithms in volume, but supporting different tracking methods requires an expanding number of ASICs, leading to larger devices, greater power consumption, and higher cost. GPUs also allow construction of economical and parallel architectures. Several processor intensive algorithms including contrast enhancement, color conversion, edge detection, feature point tracking, etc. can be offloaded to GPUs.

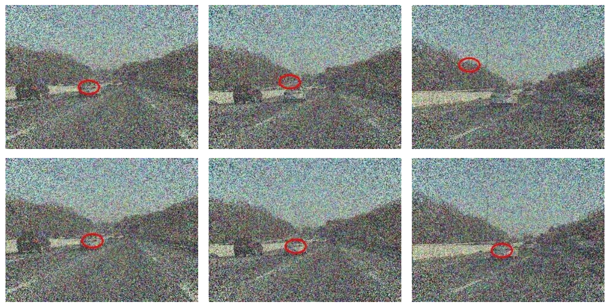

Fig. 7. Noise performance for the frames 1 (left), 40 (middle), and 200 (right). **Top:** Mean-shift tracker [?] using color histogram. **Bottom:** Covariance tracker [?] using 7 features. Almost 95% of pixels are distorted by noise.

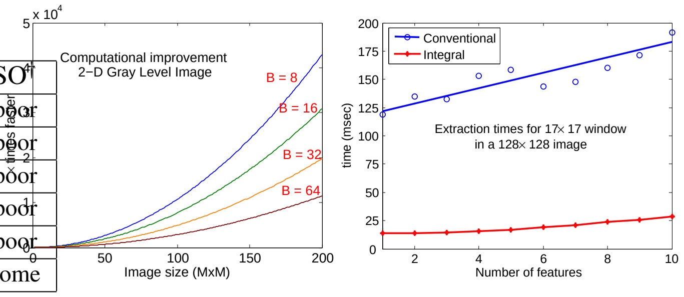

Fig. 8. **Left:** Extraction of histograms is accelerated by using integral histograms [?]. Each graph corresponds to a different histogram size. **Right:** Covariance extraction also improves by using the integral images. Covariance is computed for 17×17 image windows.

FPGAs enables large-scale parallel processing and pipelining of data flow. Latest FPGAs provide significant on-chip RAM and support high clock speeds. However, current on-chip RAMs are not sufficient to support a useful level of internal RAM frame buffering in object detection and tracking. Therefore, additional external memory banks are required to provide storage during processing of image data. The high I/O capability of FPGAs supports access to multiple RAM banks simultaneously, enabling effective and efficient pipelining. By using multiple memory banks, background subtraction for a 512×512 frame takes only 1.96msec on a cheaper 100Mhz FPGA, whereas the same operation requires 77.4msec for a 30Mhz TMS320C44 and 3.15msec on a 1.1GHz TMS320C64x DSP board [?].

Still, there remains need for algorithmic improvements to achieve a real-time tracking performance under uncontrolled conditions:

- Likelihood score computation between the object and the candidate regions is a bottleneck. Tracking methods employing histograms become more demanding as the histogram size increases. Some histogram distance metrics (Bhattacharya, KL) are inherently expensive. For covariance tracking, the likelihood computation requires extraction of eigenvectors, which is slow. Fast likelihood computation methods can significantly improve the computational speed.
- Complexity is proportional to the number of the candidate regions (or the search region size). Hierarchical search methods can be applied to accelerate the tracking process.
- Localized search methods such as mean-shift and ensemble tracking become slower as the object size becomes larger. Adaptive scaling of the kernels and images without destroying the salient information can be adapted to achieve a real-time performance.
- Kernel based tracking methods becomes more demanding as the number of objects increases. Global search methods can be applied for applications that require tracking of a multitude objects.

IV. CONCLUSIONS

As discussed above, object detection and tracking are among the computationally most demanding image processing tasks.

In addition to the mentioned algorithmic developments, special hardware implementations would significantly improve the speed of these tasks. Pixel-wise operations in object detection and search operations in tracking can easily be processed in parallel. On the other hand, robustness to extreme conditions requires scene adaptation and ability to select between different detectors and trackers. We expect hybrid solutions that can predict and apply the best combination of the detectors and trackers to be more popular in the future. Nevertheless, such solutions would require even more computational power.