A SURVEY OF VIDEO TRACKING ALGORITHMS WITH DIFFERENT IMPLEMENTATIONS: A COMPARATIVE APPROACH

[1]HUGO SBAI, [2]SAMY MEFTALI, [3]DJAMEL AOUALI

University Of Lille. CRIStAL Laboratory Villeneuve d'Ascq, France
Dcarte Engineering Company Paris, France
E-mail: samy.meftali@univ-lille1.fr, Djamel.Aouali@dcarte.fr

Abstract - Todays security requirements make video surveillance an absolute imperative in almost all important companies and administrations over the world. Objects and individuals tracking one of the most challenging problems in most modern surveillance system, thus many approaches and algorithms are specially proposed, in the literature, for this issue. These algorithms give different performances depending on the context of their use. In addition, the choice of software or hardware implementation has a significant influence on the accuracy, performances and complexity of the overall system. In order to make easier for designers to make the best choices and decisions to build efficient video tracking systems, we give in this paper a survey of the most significants algorithms and possible implementations.

Keywords - Video surveillance, Tracking algorithm, Software, Hardware, FPGA, Performances.

I. INTRODUCTION

The common feature of most detection and tracking algorithms is their high computational complexity due to amount of data that needs to be processed. This is especially important for live applications, such as video surveillance systems for threats detection or traffic monitoring. Recent trends in surveillance systems are smart cameras that perform video analysis on site. Performing the video analysis with a serial processor requires long computation time and consumes substantial amounts of energy. These factors limit application of such algorithms in live surveillance systems. Some of the object detection and tracking algorithms process data by performing repeated operations for deferent items. For example, an object detection algorithm may repeat the same computations for each image pixel. Such algorithms are suitable for implementation on a parallel processor. Graphics Processing Units (GPUs) are popular platforms for parallel processing, using software systems such as CUDA or OpenCL. Due to physical constraints and high energy consumption, their application was limited to desktop stations and smart cameras that are connected to a continuous source of energy.

With the significant evolution of FPGA technology in recent years and the integration of features such as dynamic and partial reconfigurations, new interesting perspectives are offered to the field of video tracking. Thus, thanks to the performances of modern FPGAs in terms of computing power and energy consumption, the implementation of tracking algorithms is no longer limited to the choice between the sequentiality of a processor and the software parallelism of a GPU. Now-a-days the designer has the choice between software or hardware implementations of his tracking algorithms. However, the choice is absolutely not trivial in most cases because of heterogeneous performances of existing algorithms, depending on the environment and the context of their use.
The objective of this work is to present an overview of the most used algorithms in the literature, then to evaluate them and compare their performances. Recent implementations targeting FPGAs are taken Into account in this study.

The rest of this paper is organized as follow. Section 2 introduces three of the tracking algorithms approaches. The most used evaluation criteria are presented in section 3. Section 4 discusses different tracking algorithms issued from the three categories. Our experiments results are presented in section 5. Section 6 is dedicated to FPGA implementations of tracking algorithms. Section 7 concludes this paper.

II. TRACKING ALGORITHMS

Implementation of parallel algorithms on embedded systems was mainly based on digital signal processors (DSP) and Field Programmable Gate Arrays (FPGA) [1]. Due to relatively low number of computing units, such solutions were limited to processing only low video resolutions and low frame rates. The situation started to change around 2014, when NVIDIA released the Tegra K1 chipset and Jetson TK1 development board. This chipset was mainly intended for Smartphone's, but the development board allowed for implementation and testing of data processing algorithms using the same programming language (CUDA) that desktop GPUs used, so it was possible to translate already existing algorithms to the embedded GPU. A second iteration of the chipset, Tegra X1, was released in 2015, also

with the development board, was aimed mainly at automotive systems. Both Tegra chipsets have a relatively high number of processing units and low energy consumption (TDP below 15 W). Parameters of the Tegra X1 chipset make it a suitable solution for embedded systems, such as smart cameras.

In the past years many interesting works aims to verify whether the computational power and energy efficiency of a mobile GPU chipset in processing video streams with high resolutions is enough for considering its implementation in smart cameras(and this also much with our project). For this purpose I have resume three computationally complex algorithms were selected for evaluation. One of the most popular approaches to object detection, namely the background subtraction algorithm based on Gaussian filter, proposed by Stau□er and Grimson [23] and later extended by Zivkovic [32], was chosen as an implementation of the object detection algorithm. The parallel nature of the algorithm makes it suitable for implementation on various parallel processing platforms, such as GPUs [20,25], FPGAs or supercomputer clusters [27].

For object tracking, different approach has to be taken for scenes with isolated moving objects (such as vehicles in a street) and for scenes with a large number of objects (person) that are impossible to isolate (e.g. spectators at a sport event). For the first case, object tracking is usually performed with Kalman filters [28], with additional algorithms for resolving conflicts when the objects merge or split [10].

The disadvantage of this approach is its low precision when a large number of tracks from moving objects cross and conflict. Another approach to tracking is based on particle filters [5,11] and it proved to be more robust in object tracking in more complex situations [14,18]. No publications on implementation of this approach on a GPU were found, however FPGA optimized approach was recently proposed by Schwiegelshohn et al. [22].

In complex scenes with a high number of moving objects, a different approach is needed. Optical flow methods [4,21,24,29] are commonly used for this task, particularly for obtaining per-pixel motion vectors in crowded scenes usually as the first image processing stage. Many implementations on various devices including FPGAs of optical flow algorithms can be found in the literature. Zach et al. [30] demonstrated variation optical flow computation on a GPU. They achieve nearly real time performance for 320×240 resolution. Improved method, also on GPU, has been proposed by Gwosdek et al. [12]. Pauwels et al. [19] implemented modified phase-based Fleet and Jepson method. However, the most popular pixel motion estimation is Lucas-Kanade method [8] with proposed GPU implementation [15]. Apart from FPGAs [17], Lucas-Kanade Optical Flow method was also evaluated on TI C66x digital signal processor [31].

2.1. The background subtraction algorithm

The background subtraction (BS) algorithm divides pixels of a single video frame into two classes: the foreground (pixels belonging to moving objects) and the background. The most popular BS algorithm is the one based on Gaussian mixture models (GMM) [23, 32], so it was chosen for the GPU implementation. The algorithm is inadequately parallel because each pixel is processed independently of the others, and it is well-suited for implementation on parallel platforms [25, 27].

The GMM algorithm can be implemented as a single GPU kernel. Each pixel is processed independently from the others, so division of data into work groups can be arbitrary (shared memory is not used). Performance of the algorithm is bound by memory access, since reading and writing the pixel models consumes most of the kernel runtime [25]. On most GPU platforms, the speed of memory operations depends on the access pattern [1]. In order to optimize the algorithm, the background model in the GPU memory is ordered by features rather than by pixels. This approach reduces the risk of memory access conflicts, ensuring that consecutive threads access adjacent memory locations. The video frame data is loaded into a 24-bit buffer in the texture memory of the GPU for faster access.

2.2. Object Tracking with Particle Filters

Particle filters are useful in object tracking mainly because they provide multiple hypotheses on the state of moving objects, as opposed to typically used Kalman filters that model a single hypothesis. As a result, the risk of losing the tracked object is reduced. A methodical description of particle filters and their applications for object tracking purposes may be found e.g. in [5,11]. The algorithm is complex in terms of processing, but its parts may be realized with parallel processing. However, its implementation on a GPU platform is much more complicated compared with the GMM algorithm. It is necessary to divide the algorithm into several stages, implemented with individual GPU kernels.

A single iteration of a particle filter consists of a prediction and an update stage. Updating the filter requires a measurement of validity of each prediction. For object tracking in video, color histograms are usually applied for computing the measurement. We can find many examples partially based on [18] and [14].

Implementation of the algorithm in CUDA was divided into several kernels. The most difficult part to implement was the update phase which involves computing a large number of histograms. Each

histogram requires allocating a memory buffer, and histogram bins are accessed in a non-sequential manner, resulting in a suboptimal memory access pattern.

2.3. Optical Flow method

The appearance representation model of the object is the foundation of any vision-based tracking algorithm. The Virtual Gate algorithm performing people counting in a mass flow is based on the Optical Flow method. The design of the algorithm assumes a top camera view of human silhouettes in places where crowd motion is expected, especially at entrances, stairways, etc. [9].

Lucas and Kanade were the first to adopt the original gray scale of images to represent the whole object through template [1,2]. In order to better track the occluded object, Mei and Ling [3] proposed a tracking algorithm based on sparse representation, which adopted the positive and negative template as well as the noise template to represent the object. In addition, there are many visual features of the object, including the color histogram [4], the gradient direction histogram (HOG) [5], the covariance region descriptor [6] and the Haar Like feature [7]. Those visual features are used in modeling the appearance of objects. On the other hand, some scholars took full advantage of the feature points of the object to detect and track objects by extracting the feature points in the object image. Those frequently-used feature points include Moravec features [8], Harris corners [9], KLT [10] and SIFT [11]. There also exist modeling methods based on subspace models, such as PCA and LDA, which are both widely adopted in the field of image tracking.

To enhance the visual tracking robustness, Babenko et al. [8, 9] proposed a multiple instance learning (MIL) algorithm, which uses a set of positive samples and multiple sets of negative samples acquired in the neighborhood of the optimal location of the target to conduct classifier training to obtain an optimal probabilistic classifier. The trained classifier is used to detect the target on the image at the next moment. After obtaining the optimal location of the target, the classifier is updated by online sampling of new positive and negative samples to track the target in the next image. However, the MIL algorithm does not consider the importance of each sample and treats all samples as having the same importance. In practice, positive samples are sampled within a specific region around the center of the target ROI.

III. TRACKING EVALUATION MEASURES

Many measures for evaluating the tracking performance have been proposed, typically with the comparison against ground truth, considering the target presence and position. This requires a considerable amount of annotation, with the consequence that the amount of videos with ground truth is often limited up to this point. [33] proposed performance measures without ground truth by evaluating shape and color differences of the results.

The Performance Evaluation of Tracking and Surveillance, PETS, workshop series was one of the first to evaluate trackers with ground truth, proposing performance measures for comparing tracking algorithms. Other performance measures for tracking are proposed by [38] and [24], as well as in [36]. In the more recent PETS series, [29], VACE [39], and CLEAR [40] metrics were developed for evaluating the performance of multiple target detection and tracking, while in case of single object tracking evaluation there is no consensus and many variations of the same measures are being proposed. Here we provide a survey of the most common measures used in single target tracking, as summarized in Table 1.

TABLE 1
Overview Characteristics of the Evaluation Metrics

Name	Equation	Aim	Measure
F-score [43]	$2 \cdot \frac{precision \cdot recall}{precision + recall}$	Accuracy	Thresholded precision and recall
$F1$-score [31]	$\frac{1}{N_{frames}} \sum_t 2 \cdot \frac{p^t \cdot r^t}{p^t + r^t}$	Accuracy	Precision and recall
OTA [40]	$1 - \frac{\sum_t (m_p^t + fn_p^t)}{\sum_t g^t}$	Accuracy	False positive and false negative
OTP [39]	$\frac{1}{\lvert M_c \rvert} \sum_{i \in M_c} \frac{\lvert T^i \cap GT^i \rvert}{\lvert T^i \cup GT^i \rvert}$	Accuracy	Average overlap over matched frames
ATA [31]	$\frac{1}{N_{frames}} \sum_t \frac{\lvert T^t \cap GT^t \rvert}{\lvert T^t \cup GT^t \rvert}$	Accuracy	Average overlap
$Deviation$ [47]	$1 - \frac{\sum_{i \in M_c} d(T^i, GT^i)}{\lvert M_c \rvert}$	Location	Centroid normalized distance
PBM [31]	$\frac{1}{N_{frames}} \sum_t \left[1 - \frac{Distance(i)}{T_h(i)}\right]$	Location	Centroid L1-distance

Table 1. Overview characteristics of the evaluation metrics

In this survey we aimed to include trackers from as diverse origin as possible to cover the current problem. We have selected several trackers with an established reputation as demonstrated by the number of times they have been cited and reused, supplemented with trackers which have appeared in the major research works during the recent last years.

IV. DETAILED CLASSIFICATION OF THE TRACKING ALGORITHMS

4.1. The background subtraction algorithm

The trackers in the first group perform a matching of the representation of the target model built from the previous frame(s).

[NCC] Normalized Cross-Correlation: A most basic concept of tracking is direct target matching by normalized cross-correlation which uses the intensity values in the initial target bounding box as template, in [49] where a fast algorithm is presented. At each frame, the tracker samples candidate windows uniformly around the previous target position. Each candidate window is compared against the target template using normalized cross correlation. The candidate with the highest score is selected as the new target location. In this version of NCC, no updating of the target template takes place.

[KLT] Lucas-Kanade Tracker: Ahead of his time by a wide margin, the tracker finds the affine-transformed match between the target bounding box and candidate windows around the previous location. The affine transformation is computed by incremental image alignment based on spatiotemporal derivatives and warping capable of dealing with scale, rotation and translation. The location of the target is determined by mapping the target position in the previous frame to the location in the current frame using computed affine transformation [50].

[KAT] Kalman Appearance Tracker: The paper in [51] addresses appearance change by appearance-predicted matching for handling target change under occlusion. The target region is represented by 20x20 template intensities each of which is associated with a separate Kalman filter (with the parameters pooled over all predictors). In an additive Gaussian noise model, the filter predicts the development of each template-intensity over time. The target motion is modeled by a 2-D translation at a single scale and searched in an area around the previous target position. In the sub sequent frame, candidate windows around the predicted target position are reduced to a 20×20 template and compared with the predicted template, while penalizing large differences to reduce the effect of outliers. The candidate with the lowest overall difference is selected. In KAT, the template is updated with the newly (predicted) target.

[FRT] Fragments-based Robust Tracking: The tracker in [52] pursues matching the ensemble of patches the target bounding box is broken into. In this way partial occlusions and pose changes can be handled patch-by-patch. A fixed array of 10 by 2 concatenated patches, here called fragments, keep track of the changes. When a new frame [MST] Mean Shift Tracking: The famous tracker [53] performs matching with histograms rather than using any spatial information about the pixels, making it suited for radical shape changes of the target. It represents the target by an RGB-color histogram. For each new frame, the tracker compares candidate windows with the target region on the basis of the metric between their histograms. MST uses the target histogram formed in the first frame without any update during tracking.

[LOT] Locally Orderless Tracking: The tracker [42] offers adaptation in object appearance by matching with flexible rigidity. Given the initial bounding box, the target is segmented into super pixels. Each super pixel is represented by the center of mass and average HSV-values. The target's state is sampled using a Particle Filter with a Gaussian weight around the previous position. Each particle corresponds to a candidate window for which the super pixels are formed. The likelihood of each window is derived from a parameterized Earth Mover's Distance (see also FRT above) between the super pixels of the candidate and the target windows where the parameters determine the flexibility of the target. The new target state is the likelihood-weighted sum over all windows. Updating is done via the parameters of the noise model and the parameters of the EMD.

4.2. Object Tracking with Particle Filters

An essential step forward, for long term tracking especially, was the idea of maintaining an extended model of the target's appearance or behavior over the previous frames. This comes at the expense of having to search for the best match both in the image and in the extended model of appearance variations.

[IVT] Incremental Visual Tracking: The tracker in [54] recognizes that in tracking it is important to keep an extended model of appearances capturing the full range of appearances of the target in the past

[TAG] Tracking on the Affine Group: The paper [56] also uses an extended model of appearances. It extends the traditional {translation, scale, rotation} motion types to a more general 2-dimensional affine matrix group. The tracker departs from the extended model of IVT adopting its appearance model including the incremental PCA of the target intensity values. The tracker samples all possible transformations of the target from the affine group using a Gaussian model.

[TMC] Tracking by Monte Carlo sampling: The method [43] aims to track targets for which the object shape changes drastically over time by sparse optimization over patch pairs. Given the target location in the first frame, the target is modeled by sampling a fixed number of target patches that are described by edge features and color histograms.

[ACT] Adaptive Coupled-layer Tracking: The recent tracker [57] aims for rapid and significant appearance changes by sparse optimization in two layers. The tracker constraint changes in the local layers by maintaining a global layer. In each local layer, at the start, patches will receive uniform weight and be grouped in a regular grid within the target bounding box. Each layer is a gray level histogram and location. For a new frame, the locations of the patches are predicted by a constant-velocity Kalman-filter and tuned to its position in the new frame by an affine transformation. Patches which drift away from the target

[L1T] L1-minimization Tracker: The tracker [58] employs sparse optimization by L1 from the past appearance It starts using the intensity values in target windows sampled near the target as the bases

for a sparse representation. Individual, non-target intensity values are used as alternative bases. Candidate windows in the new frame are sampled from a Gaussian distribution centered at the previous target position by Particle Filtering. They are expressed as a linear combination of these sparse bases by L1-minimization such that many of the coefficients are zero.

[L1O] L1 Tracker with Occlusion detection: Advancing the sparse optimization by L1, the paper [59] uses L2 least squares optimization to improve the speed. It also considers occlusion explicitly. The candidate windows are sorted on the basis of the reconstruction error in the least squares. The ones above a threshold are selected for L1-minimization. To detect occluded pixels, the tracker considers the coefficients of the alternative bases over a certain threshold to find pixels under occlusion. When more than 30% of the pixels are occluded, L1O declares occlusion, which disables the model updating.

4.3. Optical Flow method

A different view of tracking is to build the model on the distinction of the target foreground against the background. Tracking-by-detection, as it is called, builds a classifier to distinguish target pixels from the background pixels, and updates the classifier by new samples coming in.

[FBT] Foreground-Background Tracker: A simple approach to the incremental discriminative classifier is [60] where a linear discriminate classifier is trained on Gabor texture feature vectors from the target region against feature vectors derived from the local background, surrounding the target. The target is searched in a window centered at the previous location. The highest classification score determines the new position of the target in FBT. Updating is done by a leaking memory on the training data derived from old and new points in the target and in the surrounding. We use the version in [61] with color SURF features rather than intensities as in the reference.

[HBT] Hough-Based Tracking: The recent tracker [62] aims at tracking non-rigid targets in a discriminative classifier with segmentation of the target. A rectangular bounding box will introduce many errors in the target/background labels into the supervised classifier, especially for non-rigid and articulated targets. Therefore, the authors aim to locate the support of the target through back projection from a Hough Forest.

[SPT] Super Pixel tracking: The recent method [66] embeds the discriminative classifier in super pixel clustering. The purpose is to handle changes in scale, motion, and shape with occlusion. The HSI-histograms of the super pixels are extracted in the first 4 frames from the extended target region. Using mean-shift clustering, super pixels are grouped on the basis of the super pixel histograms with a cluster confidence derived from the overlap of the cluster with the target bounding box. In the new frame, candidate windows are sampled weighed according to a Gaussian distribution around the previous location. The search space is enlarged by considering different scales of the candidate windows.

[MIT] Multiple Instances learning Tracking: The paper [46] recognizes the difficulty in taking the current tracker region as the source for positive samples and the surrounding as the source for negative samples as the target may not completely fill the bounding box or cover some of the background.

[TLD] Tracking, Learning and Detection: The paper in [1] aims at using labeled and unlabeled examples for discriminative classifier learning. The method is applied to tracking by combining the results of a detector and an optical flow tracker. Given the target bounding box in the first frame, the detector learns an appearance model from many 2bit binary patterns [68] differentiated from patterns taken from a distant background. The authors use the fast Random Ferns [69] to learn the detector. When a new frame arrives, locations with some 50 top detector scores are selected. The optical flow tracker applies a KLT to the target region and proposes a target window in the current frame. The normalized cross correlation is computed for the candidate windows. The system selects the candidate window which has the highest similarity to the object model as the new object. Once the target is localized, positive samples are selected in and around the target and negative samples are selected at further a distance to update the detector target model. If neither of the two trackers outputs a window, TLD declares loss of target. In this way TLD can effectively handle short-term occlusion.

[STR] STRuck: Structured output tracking with kernels: The structured supervised classifier [70] circumvents the acquisition of positively and negatively labeled data altogether, as it integrates the labeling procedure into the learner in a common framework. Given the target bounding box, it considers different windows by translation in the frame.

V. EXPERIMENTATIONS AND ANALYSIS

5.1. Parameters that influence the trackers

A. Illumination Conditions

An important aspect of illumination is the appearance of a shadow in the path of the object. This can be overcome by keeping a model of the

target as well as of the local background. Figure 1 illustrates the appearance of shadows for all the compared algorithms.

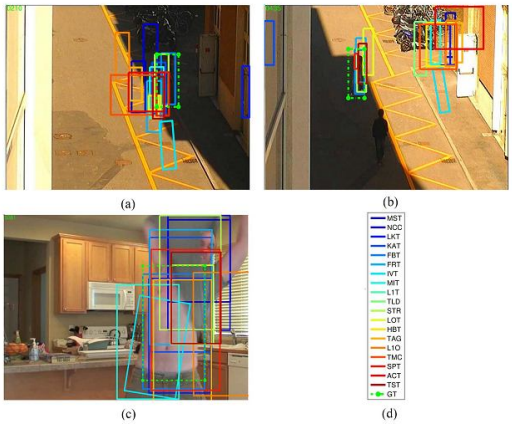

Figure 1. Shadow effects

B. Changes in the Target's Surface Cover

Keeping a model for the background separately from a model of the target is also advantageous when the surface cover changes drastically over time.

C. Shape and Size

In general, the performance of trackers on drastic changes in shape is poor. Of all aspects of difficulty, the overall performance on videos with shape changes is lowest. Tracker L1T which is very flexible - i.e. it places few shape constraints in its appearance model - is the best overall in this category.

D. Camera Motion

Camera motion is the one aspect of difficulty for which there is a successful strategy indeed.

E. Length of the Video

As can be seen in the figure 2 (parameter 14), the best performance is by R, FBT, NCC and TLD which are also good performers, in general. TLD is the only tracker capable of making something out of long sequence.

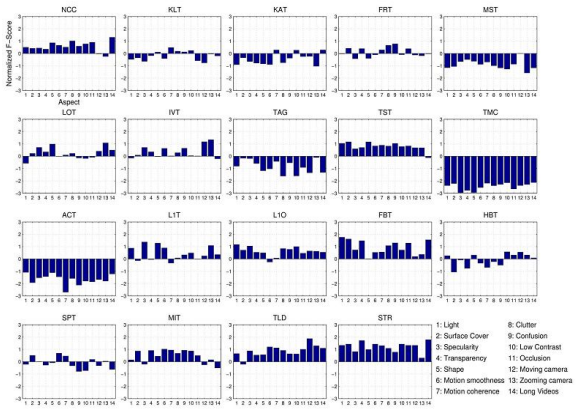

Figure 2. Normalized performance score for compared algorithms overs usual criteria

5.2. The Performance of Trackers

Overall STR performs the best. Furthermore, it has an even performance on all aspects of difficulty (Fig. 3). Although it has a reasonable performance in many complex videos, it rarely achieves an outstanding performance in which it tracks significantly better than any of the other trackers. The solid performance is attributed to the precisely motivated use of S-SVM which optimizes the displacement directly in a discriminative setting. It only fails on scale changes of the target as it has no mechanism to detect that.

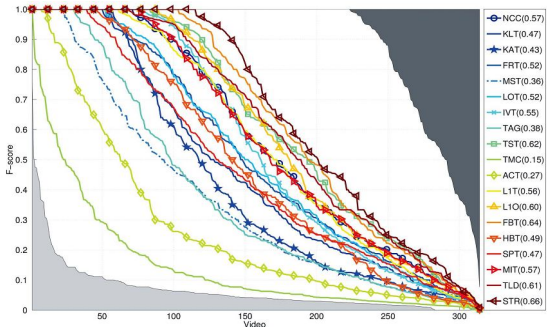

Figure 3. Performances of tracking algorithms (F-score)

TLD performs remarkably well on camera motion due to
its well-designed detection and motion model. The number of outstanding performances in occlusion and camera motion is much larger than any other tracker. Also the overall performance is good. Only in categories related to illumination and appearance the performance is limited. FBT has good overall performance, while specializing in appearance changes of the target and changes in illumination. It does not know how to handle changes in shape or camera motions as it cannot handle the smearing of the feature values in its discriminative approach.
Good overall performance is also delivered by TST which is a collection of many weak trackers with a solid performance on all aspects; apart from long video as the model updating has too big a complexity. L10 is the improved version of L1T designed to handle occlusions, but - as we have found - it improves also on other aspects of L1T.

VI. FPGA IMPLEMENTATIONS OF TRACKING ALGORITHMS

Several FPGA architectures have been developed and several solutions have been proposed. [24, 17]. In [8] embedded system architecture for feature detection and matching was presented. The proposed FPGA architecture implements the FAST [15] (Features from Accelerated Segment Test) feature detector and the BRIEF [5] (Binary Robust Independent Elementary Features) feature descriptor in a customizable FPGA block. The developed blocks

were designed to use hardware interfaces based on the AMBA AXI4 interface protocol and were connected using DMA (DirectMemory Access) architecture. The proposed architecture computes feature matching over two consecutive HD frames coming from an external memory at 48 frames per second. In [26] a FPGA architecture of SIFT (Scale Invariant Feature Transform) visual descriptor associated to an image matching algorithm was presented. For an efficient FPGA-SIFT image matching implementation (in terms of speed and hardware resources usage), the original SIFT algorithm was optimized as follows: 1) Up sampling operations were replaced with down sampling, in order to avoid interpolation operations. 2) Only four scales with two octaves were used. 3) Dimension of the visual descriptor was reduced to 72 instead of 128 in the original SIFT formulation. This implementation is able to detect and match features in 640*480 image resolution at 33 frames per second. More recently, Weberruss [25] have proposed a FPGA architecture for ORB [16] (Oriented FAST) descriptor associated to a feature matching algorithm. "harris corner" detection [10] was the feature extractor and ORB visual descriptors were computed at each "corner". Finally, the previous features (stored in 2D Shift Register) and the current features were matched using the hamming distances as discrimination metric. In 2017, Vourvoulakis [23] presented FPGA-SIFT architecture for feature matching. In order to achieve high hardware parallelism, procedures of SIFT detection and description were reformulated. At every clock cycle, the current pixel in the pipeline is tested and if it is a SIFT feature, its descriptor is extracted. Furthermore, every detected feature in the current frame is matched with one among the stored features of the previous frame, using a moving window, without breaking the "pixel pipeline". False matches are rejected using RANSAC (Random Sample Consensus) algorithm. The architecture was implemented on Cyclone IV. Maximum supported clock frequency was set as 25 MHz and the architecture was capable to process 81 frames per second, considering 640*480 image resolution.

In most of cases, previous FPGA-based feature matching formulations provide relatively good performance under real world scenarios. Unfortunately, in several applications and in particular smart cameras applications, these algorithms are not compliant due to their relatively high hardware requirements and their algorithmic formulation. We can mention three important limitations affecting the current feature matching algorithms:

- Low performance for embedded applications: nowadays computers can process several feature matching algorithms in real-time. Unfortunately, in embedded applications such as, smart cameras, mobile applications, autonomous robotics or compact smart vision systems, the use of computers is difficult due to their high power consumption and size. The use of FPGA technology is an alternative, but there are hard challenges due to previous visual descriptors (SIFT, BIERF, ORB) and matching techniques were designed for software implementation, and often, there are several iterative operations that could not be parallelized. As result, most previous FPGA architectures have high hardware requirements and relatively low processing rate.
- Sparse matching: in order to maintaining high discrimination between descriptors only features with high thresholding response are matching (since it is assumed that these features have to be associated with high responsive visual descriptor that has low probability to be similar in other features). In practice, this assumption ensures consistency in the matching process; however, there is an important limitation because only a few image points are matched.
- Outliers: In certain cases, the image ambiguities around features (color/texture repeatability, occlusion, etc.) generate similar visual descriptors for two or more different features, in such scenario the matching techniques deliver wrong results that can affect the global performance of several computer vision applications (camera calibration, structure from motion, visual odometer, etc.), see Fig. 2. To solve this problem, statistically robust methods like RANSAC have to be applied as outlier filter.

6.1. TLD algorithm

Combining learning capabilities with a particle filter improves the robustness of a tracking system, as demonstrated in [11]. A directional-edge based similarity assessment with a template of object appearances helps tracking robustly an object of interest.

The TLD algorithm [3] deals with the tracking problems via a combination of three main components: tracking, learning and detection. The interaction between these components adds a degree of intelligence to the classical object tracking tasks.

The TLD algorithm (Figure 4) uses a median flow tracker [13]. The median flow tracker is a technique based on a grid of points where the Kanade-Lucas-Tomasi (KLT) [14] method is applied individually to each of them. The tracking process is based on a combination of the most reliable points. The points in a bounding box, depicted in the past figure firstly tracked with KLT in order get their displacements in the next frame, these displacements being called a

forward tracking. After that, these points (with their neighborhood) are tracked inversely to the original frame to assess the capability of the algorithm to find back their original position. The points with a low forward-backward error (difference between original positions and predicted original positions) and a high similarity between original and tracked patches are considered as reliable and their tracking is used to retrieve the scale and object displacement as illustrated in Figure 1b. The remaining points are considered as outliers. The similarity between the original and predicted patches is assessed using the Normalized Cross-Correlation (NCC).

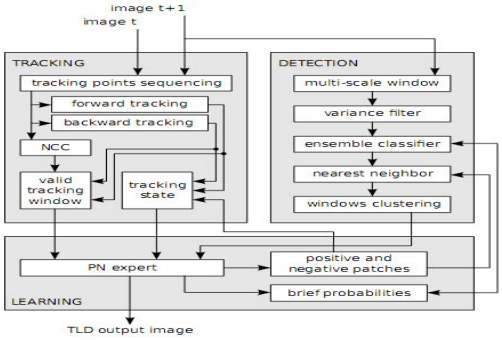

Figure 4. TLD Architecture

CONCLUSIONS

Video surveillance in general and the tracking of objects and individuals in particular is at the heart of the needs of today's companies because of the imitative of security. Thus, the development of efficient and reliable tracking algorithms and architectures that can execute them efficiently have become challenges of great importance for the scientific community. However, this has spawned a multitude of approaches and algorithms dedicated to tracking as well as potential execution targets ranging from sequential processor to FPGAs via GPUs.

In this paper we have presented the main tracking algorithms existing in the literature today. These algorithms were then evaluated and compared on predetermined criteria. Particular attention has been given to taking into account the various possible execution platforms.

REFERENCES

[1] H. Nam and B. Han, "Learning multi-domain convolutional neural networks for visual tracking," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4293–4302.

[2] M. Tomasi, S. Pundlik, and G. Luo. Fpga–dsp co-processing for feature tracking in smart video sensors. Journal of Real-Time Image Processing, 11(4):751–767, 2016.

[3] J. Vourvoulakis, J. Kalomiros, and J. Lygouras. Fpga accelerator for real-time sift matching with ransac support. Microprocessors and Microsystems, 49:105–116, 2017.

[4] Ranran, L., Enxing, Z., Shan, C., Shaoyi, B., Lanchun, Z.: Hierarchical stochastic gradient identification for non-uniformly sampling Hammerstein systems with colored noise. Comput. Syst. Sci. Eng. 31(6), 425–430 (2016)

[5] Liu, R., Xu, H., Zheng, E., et al.: Adaptive filtering for intelligent sensing speech based on multi-rate LMS algorithm. Clust. Comput 20(2), 1493–1509 (2017)

[6] Liu, R., Pan, T., Li, Z.: Multi-model recursive identification for nonlinear systems with non-uniformly sampling. Clust. Comput. 20(1), 25–32 (2017)

[7] Divvala, S.K.: An empirical study of context in object detection. ComputerVision and PatternRecognition,CVPR2009. IEEEConference on IEEE, pp. 1271–1278 (2009)

[8] Adarve, J.D., Mahony, R.: A filter formulation for computing real time optical flow. IEEE Robot. Autom. Lett. 1(2), 1192–1199 (2016)

[9] Czy˙ zewski, A., Bratoszewski, P., Ciarkowski, A., Cichowski, J., Lisowski, K., Szczodrak, M., Szwoch, G., Krawczyk, H.: Massive surveillance data processing with supercomputing cluster. Inf. Sci. 296, 322–344 (2015)

[10] Sun, D., Roth, S., Black, M.J.: A quantitative analysis of current practices in optical flow estimation and the principles behind them. Int. J. Comput. Vis. 106(2), 115–137 (2014)

[11] Szwoch, G.: Performance evaluation of parallel background subtraction on GPU platforms. Elektronika: konstrukcje, technologie, zastosowania 56(4), 23–27 (2015)

[12] Szwoch, G.: Performance evaluation of the parallel object tracking algorithm employing the particle filter. In: 2016 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA), pp. 119–124. SPA, September 2016

[13] C. Erdem, B. Sankur, and A. M. Tekalp, "Performance measures for video object segmentation and tracking," IEEE Trans. Image Process., vol. 13, no. 7, pp. 937–951, Jul. 2004.

[14] S. Salti, A. Cavallaro, and L. di Stefano, "Adaptive appearance modeling for video tracking: Survey and evaluation," IEEE Trans. Image Process., vol. 21, no. 10, pp. 4334–4348, Oct. 2012.

[15] J. C. SanMiguel, A. Cavallaro, and J. M. Martinez, "Adaptive on-line performance evaluation of video trackers," IEEE Trans. Image Process., vol. 21, no. 5, pp. 1828–1837, May 2012.

[16] T. Nghiem, F. Bremond, M. Thonnat, and V. Valentin, "Etiseo, performance evaluation for video surveillance systems," in Proc.AVSS, London, U.K., 2007, pp. 476–481.

[17] P. Carvalho, J. S. Cardoso, and L. Corte-Real, "Filling the gap in quality assessment of video object tracking," IVC,vol.30,no.9, pp. 630–640, 2012.

[18] F. Bashir and F. Porikli, "Performance evaluation of object detec- tion and tracking systems," in Proc. IEEE Int. Workshop PETS, 2006.

[19] R. Kasturi et al., "Framework for performance evaluation of face, text, and vehicle detection and tracking in video: Data, metrics, and protocol," IEEE Trans. Pattern Anal. Mach. Intell., vol. 31, no. 2, pp. 319–336, Feb. 2009.

[20] K. Bernardin and R. Stiefelhagen, "Evaluating multiple object tracking performance: The clear MOT metrics," EURASIP J. IVP, vol. 2008, no. 1, p. 246309, Feb. 2008.

★★★