www.editada.org

International Journal of Combinatorial Optimization Problems and Informatics, 15(5), Dec 2024, 127-145. ISSN: 2007-1558. https://doi.org/10.61467/2007.1558.2024.v15i5.571

A Review of Design Methodologies and Evaluation Techniques for FPGA-Based Visual Object Tracking Systems

Víctor Alejandro Méndez López[1], Carlos Soubervielle Montalvo[1], Alberto Salvador Núñez Varela[1], Oscar Ernesto Pérez Cham[2], Emilio Jorge González Galván[1]
[1] Universidad Autónoma de San Luis Potosí, Facultad de Ingeniería, México.
[2] Universidad del Mar Campus Puerto Escondido, Juquila, Oaxaca, México.

alejandromdzlpz.uaslp@gmail.com,carlos.soubervielle@uaslp.mx,alberto.nunez@uaslp.mx, operezcham@zicatela.umar.mx,egonzale@uaslp.mx

Abstract. In recent years, computer vision algorithms have improved from conventional image processing to deep learning approaches. Meanwhile, complex but flexible FPGA-based platforms have made possible the development of challenging real-time heterogeneous systems for visual object tracking (VOT). This study presents a comprehensive review of design methodologies, algorithms, and evaluation techniques of 21 FPGA-based VOT systems reported in literature from 2017 to 2023. Five design methodology categories are described: Region matching, Feature matching, Machine learning, Deep learning, and Hybrid systems. FPGA, SoC-FPGA and MPSoC platforms for VOT system implementations are considered. Relevant evaluation techniques and metrics are reviewed as part of dataset benchmarks or toy datasets. In order to propose an insight of the FPGA-based VOT systems, each topic presents their comparative analysis and discussion. Finally, main conclusions, recommendations and perspectives are presented.

Keywords: Visual Object Tracking, FPGA-based platform, Design Methodologies, Co-design, Evaluation techniques, Datasets.

Article Info
Received May 10, 2024.
Accepted Nov 20, 2024.

1 Introduction

Visual Object Tracking (VOT) is a research field in computer vision aimed at monitoring moving objects within sequences of video images. Although this is a challenging task, in recent years the general performance of VOT systems has improved significantly due to the convergence of different image processing algorithms and hardware platforms.

The design and implementation of FPGA-based VOT systems faces several challenges, including real-time processing requirements, accuracy, robustness to variations in illumination and occlusion, changing appearance, among others [1]. In most cases, this tends towards the development of complex VOT systems with latency, area, and power restrictions that only can be compensated by using hardware and software co-design methodologies [2].

The fundamental aspect of VOT systems lies in the computational approach used to determine the optimal position and/or condition of the target in the latest frame [3]. For this, a typical VOT system [2] is composed of

five generic functional procedures (see Figure 1): a Visual Sensor or equivalent source of data to provide video frames, a Preprocessing stage for image conversion and formatting, noise reduction, etc. (if required), an Object detection stage to manual or automatic object localization in the first frame, and finally the Feature Extraction and Object Tracking modules for the main VOT task [2].

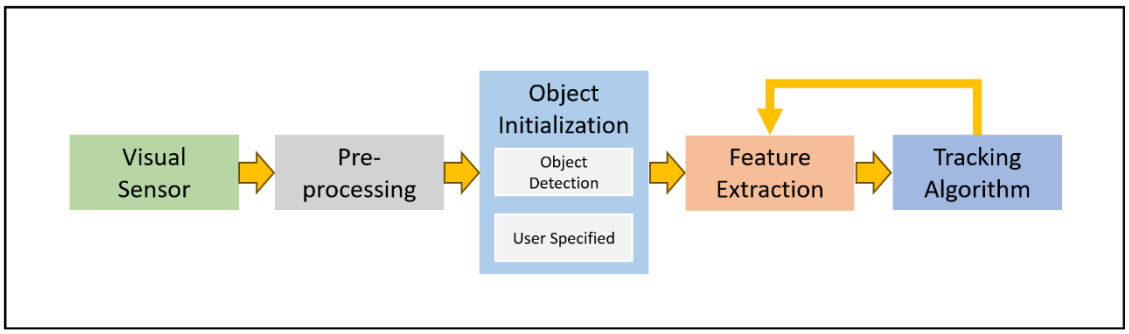

Figure 1. Typical VOT system model [2].

VOT system platforms goes from conventional workstations based on Central Processing Units (CPU) and Graphic Processing Units (GPU) to more specialized boards: Field Programmable Gate Array (FPGA), System-on-Chip FPGA (SoC-FPGA), Multi-Processor System-on-Chip (MPSoC) and Application-Specific Integrated Circuit (ASIC) [4]. In particular, an FPGA is a semiconductor device used for the development of highly reconfigurable digital logic and VLSI circuits, SoC-FPGA is the integration of a conventional CPU and an FPGA in the same silicon, and MPSoC is the result of multiple CPU cores, such as GPU and Reconfigurable Processing Units (RPU), with at least one FPGA with other advanced modules on a single chip [5]. For these FPGA-based heterogeneous architectures, Processing System (PS) and Programmable Logic (PL) refers to the internal CPU-based and FPGA-based components respectively. This paper will focus on the analysis of VOT implementations reported on these three FPGA-based platforms.

In this paper, a review of design methodologies, system implementations, and evaluation techniques for FPGA-based VOT systems is reported. For this, the rest of this paper is organized as follows: first, a brief review of related work is presented in section 2, followed by a description of the methodology applied for this review. In sections 4, 5 and 6, three research questions are presented with their corresponding results and discussions Finally, the conclusions and perspectives of this review are mentioned in section 7.

2 Related Work

The following literature offers a valuable insight into the field of VOT and reveals the dynamic landscape of VOT systems and their applications, providing a comprehensive understanding of hardware acceleration in computer vision applications.

El-Shafie et al. [2] conduct a thorough literature survey spanning two decades, primarily focusing on hardware implementations of object trackers. Their categorization of VOT systems into Mean-Shift, Filtering techniques, Feature matching, Optical flow, Template matching, and Bio-inspired based trackers underscores the diversity of approaches in this domain. Li et al. [6] conducts an investigation into various methods for achieving high-quality object tracking by examining the principles, evolutionary pathways, and recent advancements of these approaches. These methods are organized into four distinct categories: Template matching, Filtering, Classification and fusion Trackers. Fiaz et al. [7] offer a comprehensive comparative review of tracking algorithms, with a specific emphasis on the robustness of different trackers concerning feature extraction methods. They categorize trackers into Correlation Filter based Trackers (CFTs) and Non-CFTs and evaluate their performance accuracy, further dissecting them based on architecture and tracking mechanisms. Zhang et al. [8] provide a concise overview of single-object tracking algorithms developed over the past decade, and provide a VOT algorithms categorization with focus on correlation filters and deep learning techniques. Their comparative analysis evaluates the performance of these algorithms across OTB2015, VOT2016, and LaSOT datasets, highlighting the challenges of single-object tracking in dynamic and complex environments. Tiwari et al. [9] proposes a generic VOT architecture for detection and tracking methods, categorizing them into Point

tracking, Kernel tracking, and Silhouette tracking. This classification provides a structured framework for understanding different tracking approaches and their applicability in diverse scenarios.

Regarding impact of deep learning technologies, Zeng et al. [10] introduces and compares existing hardware accelerators, outlines typical deep learning-based object detectors, and elucidates the rationale behind selecting FPGA as an accelerator platform. By discussing design goals and methods for FPGA accelerators, they offer insights into optimizing performance and efficiency in object detection tasks. Feng et al. [11] highlight the transformative impact of deep learning methodologies on tasks such as image classification, object detection, and image segmentation. Their focus lies in optimizing these algorithms for real-time processing and energy efficiency across diverse hardware accelerators like GPUs, FPGAs, and emerging technologies. Seng et al. [12] emphasize the potential of FPGA technology in embedding intelligent decision-making algorithms into mobile and embedded systems. They discuss the utilization of techniques like machine learning and neural networks for applications such as object detection and surveillance monitoring.

A special mention deserves Molina et al. [5] work. They encompass models, methodologies, and frameworks for metric estimation, design space exploration, and power consumption assessment on SoC-FPGA platforms. Through a detailed analysis of features, limitations, and trade-offs, they underscore the integration of these accelerators across various research domains.

Overall, although the cited papers differ in their emphases, methodologies, and categorizations, they all underscore the importance of object detection and tracking in computer vision research, emphasizing the need for robust performance measurement of trackers based on hardware to allow meaningful comparisons between implementations. Moreover, they offer information on trends and future directions in this field of research.

Despite the potential of the VOT research field, previous literature suggests more articles are needed on the aforementioned topics. This review is presented as an effort in this direction. The main contribution of this work is to provide a comprehensive overview of the design methodologies, system implementations, and evaluation techniques for FPGA-Based VOT systems. For this purpose, five categories of algorithm-based design methodologies are presented: Region matching, Feature matching, Machine learning, Deep learning, and Hybrid systems.

In addition, a comparative analysis of the 21 selected articles is carried out in terms of design methodologies, hardware implementations and evaluation techniques to propose an insight of the main perspectives of this research field. Finally, a series of conclusions and perspectives are presented.

3 Methodology

In order to provide insight into FPGA-based design methodologies for VOT and their implementations, 21 journal papers from ACM, IEEE, Web of Science, Springer, and MDPI databases between 2017 and 2023 were selected, with 2021 as the most prolific year with 6 publications (see Figure 2). The authors of this study consider that the relatively low number of publications found is explained by the complex and multidisciplinary nature of the study topic as well as the availability of specialized human resources and materials to address this type of research.

The paper selection process for this study is based on the following criteria:

a) Paper must report a fully functional FPGA-based single or multiple VOT system. This includes articles reporting CPU/GPU/ASIC development as a complementary activity.
b) Paper must describe the VOT model and the algorithm-based design methodologies used.
c) Paper must describe the FPGA-based design and their implementation.
d) Paper must describe the evaluation techniques used.
e) System video input must be from a file or a conventional RGB camera. Due to substantial changes in architecture, systems based on event cameras or similar were declined.

The 21 selected papers were used for a comparative analysis and discussion in order to answer the following questions:

a) *What are the main FPGA-based design methodologies used for VOT systems?* This question explores the various methodologies and strategies adopted to understand the fundamental approaches and techniques employed in designing VOT systems utilizing FPGAs.
b) *How are these design methodologies applied to implement FPGA-based VOT systems?*
This question aims to delve into the processes and techniques commonly used to translate the selected algorithm-based design methodologies into functional FPGA-based VOT systems.
c) *What evaluation techniques are most used for object tracking assessment?*
The purpose of this question is to identify the evaluation techniques commonly utilized to measure the accuracy, robustness, efficiency, and other relevant characteristics of VOT systems, providing insight into the criteria and methodologies for objectively evaluating their performance.

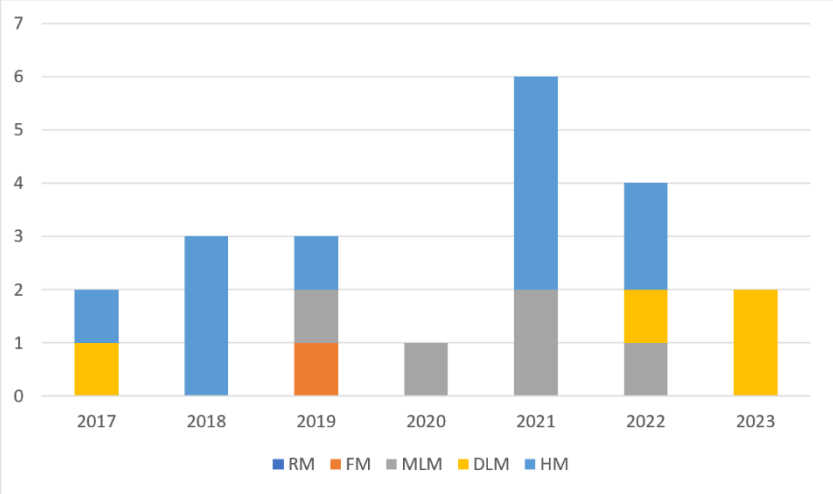

Figure 2. Reported publications per year for FPGA-based VOT systems (n=21).

4 What are the Main FPGA-based Design Methodologies Used for VOT Applications?

For the context of this question, an algorithm-based design methodology is the systematic development of a VOT implementation given an algorithm (or a set of algorithms that needs to be supported) and its specific targeted hardware platform [13]. In order to bring insight into this question, this review starts from Li et al. [6] approach to propose the following categorization (see Figure 3):

a) Region matching-based design methodologies (RMs). These algorithms search for coincidences between a template region and subsequent frames by measuring the similarity between pixel values. The new target position is determined by the location with the highest correlation score [14]. RM algorithms are often used in applications where features remain relatively stable across frames due to issues related with noise sensibility and high computational cost.
b) Feature matching-based design methodologies (FMs). These algorithms focus on extracting and matching specific features (color, shape, texture, or human features) that can be reliably detected and matched across different instances of the object [15]. FM is more suitable for object detection tasks in dynamic environments. Nevertheless, may face difficulties with complex scenes or objects with limited distinctive features.
c) Machine learning-based design methodologies (MLMs). Machine learning is an artificial intelligence paradigm in which programs learn from data to improve their performance with time when they perform some specific tasks [16]. MLM algorithms excel at handling complex scenes due to its ability to automatically learn relevant features from large amounts of data. However, the computational cost can be significant in large-scale or real-time applications.
d) Deep learning-based design methodologies (DLMs). These methodologies are based on a variety of architectures that use multiple layers of interconnected nodes, known as neurons. These architectures are designed to learn and extract hierarchical representations of meaningful data [17]. Usually, DLM implementations use lite versions of algorithms with high performance FPGA platforms due to computational resource restrictions.

e) Hybrid design methodologies (HMs). Methodologies of this category, known also as Tracker Fusion or Fusion-based trackers [6, 18], leverage the strengths of different algorithms to improve their performance. They often combine feature extraction or template matching with deep learning algorithms, capitalizing on the complementary characteristics of each technique. HM has great flexibility for leveraging characteristics of different algorithms and thus overcoming their limitations. However, its development may take longer than other design methodologies.

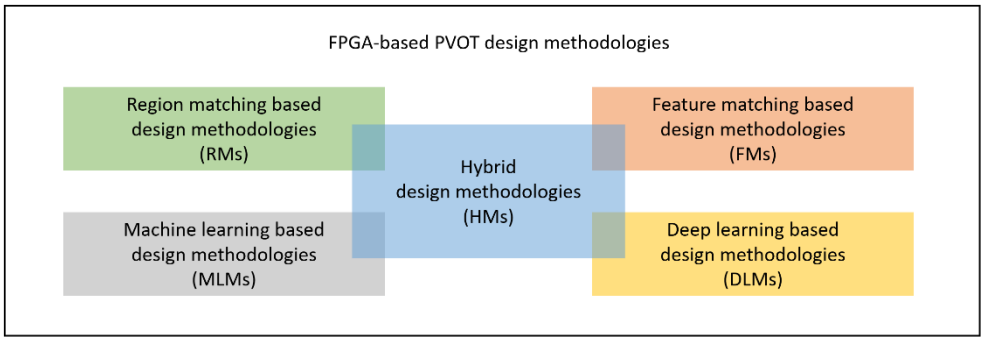

Figure 3. Proposed categorization for FPGA-based VOT design methodologies.

4.1 Results and Discussion

The following analysis is focused only on the algorithm-based design methodologies of the 21 selected papers (Figure 2). Due to Hybrid systems, which use two or more algorithms, 34 algorithms were analyzed. Although some of them are similar, we decided to preserve their original reported name. The complete list of algorithms versus design methodologies is shown in Tables 1-4. In this scenario, from the 34 reported algorithms and its variants, most reported algorithms belong to the MLM category with 12 cases (35.3%), followed by FM and DLM with 10 (29.4%) and 8 (23.5%) cases respectively. The remaining 4 cases (11.8%) are categorized as RM algorithms.

In order to analyze the use of these algorithms as part of an hybrid system, a ratio between algorithms used in hybrid systems respect total algorithms of each category is indicated as follows: the categories with highest comparative ratio are the RM and the FM category with 4/4 (100%) and 9/10 (90%) cases respectively, followed MLM and DLM with 6/11 (54.5%), and 3/8 (37.5%) respectively. In other words, all RM algorithms and most FM algorithms were implemented as part of hybrid implementations. This is explained because the algorithms of those design methodologies only perform object detection or feature extraction tasks. These particular limitations are faced and, in some cases, overcome by using two or more compatible algorithms as a part of a hybrid system. For this study, 8 of 10 papers implemented two algorithms as part of an hybrid system in one of the following methodology design combinations (number of papers in parenthesis): RM+FM (1), RM+MLM (3), MLM+DLM (2), FM only (1), FM+MLM (1). The remaining two papers used three algorithms as 2xFM+DLM (1) and 3xFM (1).

Following algorithms were found as part of non-hybrid systems exclusively: Walid et al. [19] used Discriminative scale space tracking algorithm (MLM category). It was designed carefully at core level to improve its overall real-time performance. El-Shafie et al. [20] address the limitations of ILNET Convolutional neural network algorithm (DLM category) by using interpolation schemes while facing their hardware restrictions by using a fixed-point hardware acceleration. Huang et al. [21] propose a high-speed object tracking system based on a hardware image compression circuit. They use the Self-organizing map neural network (DLM category) for the on-chip learning of the codebook's compression algorithm.

Table 1. Reported algorithms for the RM design methodology (the * symbol indicates it is used as HM)

Algorithm	Authors
Background subtraction*	Carrizosa-Corral et al. [34]
Fast background subtraction*	Montero et al. [35]
Subpixel refinement*	Aguilar-González et al. [36]

Sum of Absolute Differences (SAD)*	Singh et al. [28]

Table 2. Reported algorithms for the FM design methodology (the * symbol indicates it is used as HM)

Algorithm	Authors
Binary Robust Invariant Scalable Keypoints (BRISK)*	Nam et al. [37]
Concurrent Zero Mean Normalized Cross-Correlation (ZNCC)	Chen et al. [32]
Haar-like features*	Wu et al. [31]
Harris corner detection*	Nam et al. [37]
Local Binary Pattern Histogram (LBPH)*	Wu et al. [31]
Non-textured corner filtering*	Aguilar-González et al. [36]
Simple elliptic matching*	He et al. [38]
Spatial 2-D difference of Gaussian (DoG) filtering*	He et al. [38]
Temporal high-pass filtering*	He et al. [38]
Zero Mean Normalized Cross-Correlation (ZNCC)*	Soubervielle-Montalvo et al. [33]

Table 3. Reported algorithms for the MLM design methodology (the * symbol indicates it is used as **HM)**

Algorithm	Authors
Adaptive Block Learning (FBS-ABL)*	Montero et al. [35]
Discriminative scale space tracking (DSST) algorithm	Walid et al. [19]
Honeybee search algorithm (HSA)*	Soubervielle-Montalvo et al. [33]
Independent component analysis (ICA)*	Carrizosa-Corral et al. [34]
Kalman filter*	Iqbal et al. [23]
Kernelized correlation filters (KCF)*	Ji et al. [39]
Mean shift algorithm	Pandey et al. [24]
Mean shift algorithm	Tehreem et al. [25]
Mean shift algorithm*	Yang et al. [26]
Multidimensional Kalman filter	Babu et al. [22]
Particle filter*	Singh et al. [28]
Scalable particle filter	Engineer et al. [27]

Table 4. Reported algorithms for the DLM design methodology (the * symbol indicates it is used as **HM)**

Algorithm	Authors
Attractor neural network*	Yang et al. [26]
Deep CNN single-shot multibox detector (SSD)*	Ji et al. [39]
ILNET convolutional neural network	El-Shafie et al. [20]
Neural network-based Efficient convolutional operators (ECO)*	Iqbal et al. [23]
Self-organizing map (SOM) neural network	Huang et al. [21]
You Only Look Once (YOLOv3) CNN	Cittadini et al. [29], Zhai et al. [30]
You Only Look Once (YOLOv4 tiny) CNN*	Wu et al. [31]

Following algorithms were used for both non-hybrid and hybrid systems: Babu et al. [22] propose a multidimensional Kalman filter (KF) for linear systems with updated state vector and covariance equations. Pandey et al. [24] reports a Mean shift algorithm (MSA) implementation based on fixed-point binary logarithmic and antilogarithmic units, Tehreem et al. [25] reports an optimized fast version of MSA based on use of no fractional bits. Engineer et al. [27] propose a VOT improved parallel scalable implementation of a Particle filter (PF) algorithm for a network-on-chip platform. Cittadini et al. [29] presents a hypervisor-based VOT application to implement and evaluate a "You Only Look Once" convolutional neural network (known as YOLOv3) algorithm as the main component of a cyber-physical system, Zhai et al. [30] uses compressed YOLOv3 and YOLOv3-tiny models with fast convolution engines to implement a scalable vehicle detector.

Chen et al. [32] presents a non-hybrid concurrent Zero Mean Normalized Cross-Correlation (ZNCC) template matching hardware core designed for eye-to-hand robotic visual tracking.

Finally, the following algorithms were reported only for hybrid systems (HM category exclusively): Iqbal et al. [23] offers a comprehensive investigation into adaptive subsampling techniques with seven combinations of object detectors with KF as ROI predictors. Yang et al. [26] implement a system with a MSA algorithm and an Attractor neural network (ANN) architecture to address challenges related to occlusion and clutter in visual environments. Singh et al. [28] implemented a Particle filter of 121 particles in addition with the Sum of Absolute Differences (SAD) algorithm as a similarity computation tool to enhance the real time performance requirements of its tracker. Wu et al. [31] implements a HM system based on Local Binary pattern histogram, Haar-like features and YOLOv4-tiny algorithms to construct an intelligent security monitoring system. Soubervielle-Montalvo et al. [33] proposes the design and implementation of an hybrid system based on the Honeybee search algorithm (HSA) metaheuristic and the ZNCC similarity measure.

5 How are these Design Methodologies Applied to Implement High Performance FPGA-based VOT Systems?

Implementing VOT systems on FPGA platforms requires a comprehensive approach to take advantage of the limited hardware resources required for the implementation of sophisticated computer vision algorithms. To address the hardware acceleration design and implementation stages, researchers propose different design approaches based on one or more of the following system-level methodologies: hardware/software co-design, top-down, and bottom-up. HW/SW co-design refers to a family of design methodologies utilized in complex electronic systems to leverage and improve the mutual benefits of hardware and software design ([33, 40]). Other conventional system-level methodologies are top-down and bottom-up approaches. The top-down approach starts from a high-level perspective and ends with smallest specific details. On the other hand, bottom-up starts from the design and integration of small components to obtain the complete system. These methodologies can be used separately or combined for each of the developed modules of the system.

Most of the reviewed articles in this review present a research workflow consisting on the following stages:

a) A PC-based modeling and testing of the VOT algorithm. This stage involves following steps:
- Choose the appropriate VOT algorithm based on the requirements of the application, such as speed, accuracy, robustness, etc.
- Implement the selected VOT algorithm in any high-level programming language.
- Run the implemented VOT algorithm on the dataset to track objects in the video sequences. Evaluate the algorithm's performance metrics such as tracking accuracy, speed, robustness to occlusions and appearance changes, etc.
- Fine-tune the algorithm parameters to improve its performance based on the testing results. This may involve adjusting thresholds, feature selection, or optimization techniques.
- Evaluate the performance of the VOT algorithm using benchmark datasets or real-world scenarios to ensure its effectiveness and reliability.

b) The design and implementation to match the model to the target FPGA platform. For this stage, the use of HW/SW co-design methodology for the following procedure is recommended ([33, 40]):
- Select an FPGA-based platform based on the requirements of the VOT system, considering factors such as processing power, available computing resources, power consumption, and cost.
- Analyze the VOT algorithm to identify modules suitable for hardware acceleration on the FPGA, and modules that can remain in software running on the CPU
- Map the algorithm components identified for hardware acceleration that can be implemented using logic cells, DSP blocks, and memory resources. This involves designing hardware modules (IP cores) for implementing the algorithm efficiently on the FPGA.
- Design interfaces between the FPGA and the CPU to facilitate data exchange and control signals. This may involve using communication protocols such as PCIe, Ethernet, or USB.
- Optimize the resource utilization (e.g., logic elements, memory blocks) of the FPGA for the designed hardware modules to meet resource constraints.

- Perform power and timing analysis to ensure the designed system meets power consumption and timing constraints.

c) The evaluation of the FPGA-based VOT system. This final stage addresses following actions:
- Verify that the FPGA-based system produces accurate and real-time object tracking results consistent with the PC-based model.
- Fine-tune the FPGA design and system parameters to optimize performance metrics such as speed and resource utilization.
- Evaluate the FPGA-based system using different datasets in terms of accuracy, speed and power consumption to ensure its reliability and effectiveness.

Regarding FPGA development platforms, Xilinx (AMD) and Altera (Intel) are the main manufacturers in the high-end FPGA market. Altera was acquired in 2015 by Intel and Xilinx was acquired in 2020 by AMD. Both companies offer their customers a broad line of products including programmable devices, software design tools, and Intellectual Property (IP) hardware functions to facilitate the design and implementation of FPGA-based systems [41].

5.1 Results and Discussion

For the 21 reviewed papers of this study, different tailored hardware/software methodologies for FPGA-based VOT systems, ranging from HW/SW co-design to exclusively top-down or bottom-up are reported. Following, notable findings about use of HW/SW co-design methodology are described.

5.1.1 Reported Methodologies

He et al. [38] compare their VLSI system implementation with a HW/SW co-design based system, which yields a lower frame rate (in comparison with other systems) of 30fps for a resolution of 640×480 pixels. While the authors attribute this to the co-design implementation itself, it may not necessarily be the case, as evidenced by the comparison indicating that this system operates at a frequency of 24MHz, the lowest among all systems compared. On the other hand, Ji et al. [39] propose a co-design approach to enhance the real-time performance of the SSD algorithm. Despite the SSD model being quantized as an 8-bit fixed-point model, co-design is crucial in leveraging the advantages of ARM and FPGA fully, ensuring real-time performance without sacrificing accuracy.

Montero et al. [35] conduct a comparison to demonstrate the advantages of utilizing a compute accelerator such as an FPGA in implementing their FBS-ABL algorithm. They present two co-design-based implementations of the FBS-ABL algorithm, one for ARM Cortex only and the other for ARM+FPGA platforms. For an image resolution of 640x480, they report performance figures of 21 fps and 40 fps, respectively. Pandey et al. [24] propose an approach based on HW/SW co-design and compare it with four conventional implementations (without co-design). Their implementation achieves a frame rate of 60fps for a resolution of 640x380, outperforming the 30fps achieved by other implementations in three out of four cases. Finally, Soubervielle-Montalvo et al. [33] propose a workflow based on HW/SW co-design methodologies for designing, implementing, and evaluating a low-power embedded system for real-time video tracking. This system combines the HSA meta-heuristic with an SoC-FPGA platform, which was developed from scratch using co-design methodology.

5.1.2 Pipelining and Parallelism

As mentioned earlier, a SoC-FPGA is a highly configurable hybrid architecture that combines an ARM CPU (called processing system) and an FPGA (called programmable logic or logic fabric). For the 14 SoC-FPGA and MPSoC implementations, 8 (38.1% of total) reported detailed information of both PS and PL implementations, 3 (14.29% of total) just reported PL usage and the remaining 3 (14.29% of total) does not mention PS neither PL information. Tables 5 and 6 show tasks implemented in Processing System (PS) and Programmable Logic (PL) components.

Regarding the use of optimization techniques, this study findings reveal a diverse range of them including pipelining, parallelism, resource sharing and specific algorithmic optimizations. Although for this particular case there is not an evident difference between the use of optimization techniques for non-hybrid and hybrid VOT design methodologies, following the relevant cases are shown in the same order for comparison purposes. For the reviewed papers, pipelining and parallelism arises as the most relevant optimization techniques. Following, the identified scenarios are identified: pipelining only, parallel only and pipelining with parallel (on different modules).

For the pipelining optimization, Zhai et al. [30] implemented their DLM YOLOv3 algorithm by using a variety of optimization techniques such as memory interlayer multiplexing, parameter rearrangement, and pipelining, along with algorithmic optimizations like Im2col+GEMM and Winograd algorithms, to enhance both speed and area efficiency in their FPGA-based VOT system. On the other hand, He et al. [38] introduce a 10-stage pixel-level pipeline to facilitate pixel stream processing for his HM Simple elliptic matching algorithm, emphasizing efficient data flow and processing for real-time object tracking applications.

The parallelization only optimization main objective is to achieve real-time performance and speed up the overall tracking process. At this respect, Engineer et al. [27] focus on optimizing the execution of their MLM scalable particle filter within the region of interest (ROI) by parallelizing the process, while Singh et al. [28] adopt parallel processing elements to accelerate computation of his HM Sum of Absolute Differences (SAD), utilizing three processing elements in parallel.

Regarding the use of pipeline and parallel architectures at the same time in different modules of the system, some remarkable works focused on this efforts are Aguilar-González et al. [36] with his HM subpixel refinement algorithm implementation, Carrizosa-Corral et al. [34] with their improved HM ICA algorithm and Chen et al. [32] with his pipelining architecture for the FM Zero-Normalized Cross-Correlation (ZNCC) algorithm, as well as pipelining multi-row buffering techniques.

Table 5. Tasks implemented in Processing System (PS) component (n=14)

Platform	Task Category	Description	Authors
MPSoC	Image processing	Image acquisition, preprocessing, object tracking	[29]
	Image processing	Image capture, preprocessing, digital ROI, Kalman filter update and prediction steps via Petalinux	[23]
	Image processing	Reading and displaying of video frames, KCF tracking algorithm	[39]
SoC-FPGA	Application specific	Image files manipulation	[33]
	Image processing	HSA algorithm tasks related to intelligent coordinated decision making	[33]
	System management	Process flow control, drivers for specific operations	[34]
	System management	Feeding of video streams to particle filter via Petalinux	[27]
	System management	Image acquisition, interrupt servicing from the programmable logic	[26]
Not specified / not used PS component			[19, 22, 32, 35, 37, 38]

5.1.3 Development Tools

Regarding the manufacturer's development platforms usage, the most reported for this review are Xilinx with 16 items (76.2%) and Intel with the remaining 5 items (23.8%). Regarding FPGA-based platforms, 7 articles are FPGA(33.33%), 11 are SoC-FPGA (52.38%), and the remaining 3 articles (14.29%) are MPSoC platforms. Of the 21 articles, the most reported platform family is the Xilinx Zynq-7000 with 11 articles (52.38%). Tables 7-9 show the complete list of reported development boards sorted by FPGA family, boards, and algorithms.

Some authors report the use of additional hardware resources. In particular, Engineer et al. [27] reported the implementation of the same particle filter algorithm in Zedboard and Virtex-4 FPGA boards. For this review, only the Zedboard implementation of this author was considered. On the other hand, Zhai et al. [30] reported an NVIDIA Tesla V100 platform was used for training and pruning quantization of their YOLOv3 model, while the detection inference was implemented by the SoC-FPGA platform.

The use of programming languages is strongly dependent on manufacturer development tools and the researchers technical preferences. For the Intel boards, 2 reported the use of VHDL while the remaining 3 authors do not mention any specific language. For the case of Xilinx boards, although 4 of 16 authors do not report which language was used, the remaining 12 papers report the usage of VHDL, HLS, C, RTL and even Python languages for the PS module of SoC-FPGA.

Table 6. Tasks implemented in Programmable Logic (PL) component (n=19)

Platform	Task Category	Description	Authors
FPGA only	Algorithm acceleration	256 PE's for multiply and accumulate (MAC) operations	[20]
	Algorithm acceleration	LBP model	[31]
	Application specific	Top-level FSM, memory MUX and DRAM controller, FSMs for DRAM RD and WR operations, softmax layer	[20]
	Application specific	Camera interface, DDR2 external memory interface, camera movement controller, DVI display controller, RS232 controller	[28]
	Application specific	Controller, frequency counter, table of means, final cluster center finder	[25]
	Application specific	Linux-based hardware control logic for the servo, OV5640 camera and HDMI modules, PID algorithm implementation	[31]
	Image processing	Image preprocessing, subpixel refinement, corner detection, non-textured filterion, output construction	[36]
	Image processing	Video acquisition, image thresholding, object localization and tracking (mean shift algorithm), video display	[24]
	Image processing	Object tracking module	[28]
	Image processing	Processing entities (PEs)	[25]
MPSoC	Algorithm acceleration	YOLOv3 inference	[29]
	Algorithm acceleration	The convolution and pooling layers in SSD model	[39]
	Application specific	IMU processing, motor mixer, LIDAR processing, radio acquisition, flight controller	[29]
	Image processing	DPU (deep learning unit): image capture, preprocessing, detection and postprocessing, update and predict	[23]
SoC-FPGA	Algorithm acceleration	Reformulated FastICA expression	[34]
	Algorithm acceleration	ZNCC-based template matching module	[32]
	Algorithm acceleration	Fitness function (ZNCC)	[33]
	Algorithm acceleration	Dynamic neural network with standard floating-point arithmetic units	[26]
	Application specific	VGA display module, UART data transmission module	[32]
	Application specific	Frame buffer, particle filter implementation on a 4x4 NoC network	[27]
	Application specific	VGA controller	[37]
	Image processing	Image stream buffer module	[32]
	Image processing	Image integrator, spatiotemporal filter, feature map memory bank, feature matching block	[38]
	Image processing	Gaussian filter, the Harris corner detector module, the non-maxima suppression module, the iteration process module	[37]

Not specified / not used PL component	[19, 22, 35]

An interesting topic regarding FPGA implementation is related with the usage of MATLAB mathematical software as a debugging and evaluation tool. For the reviewed papers of this study, MATLAB is used in following scenarios:

a) As a numerical simulation analysis tool ([22, 34, 36]).
b) For development of each module, compared with different authors ([19]).
c) For the evaluation and comparison purposes of the applied metrics to the proposed algorithm ([20, 27]).
d) As a tool for the transition of the PC-based design to the FPGA implementation (using MATLAB/Simulink) ([22]).
e) For comparison of the system model among PC and FPGA-SoC implementation ([19, 34]).
f) Other reported but no specified MATLAB usage scenarios ([25, 39]).

Table 7. Reported development platforms and algorithms for FPGA (n=7)

FPGA family **Board** Algorithm(s)	**Design methodology**
Intel Cyclone IV	
Not specified	
Non-textured corner filtering	FM* [36]
Subpixel refinement	RM* [36]
Intel Stratix IV GX	
Altera DE4 Development and Education Board	
Self-organizing map (SOM) neural network	DLM [21]
Xilinx Artix-7	
Nexys-Video Artix-7 FPGA trainer board	
Haar-like features	FM* [31]
Local Binary Pattern Histogram (LBPH)	FM* [31]
You Only Look Once (YOLO v4) CNN	DLM* [31]
Xilinx Spartan-6	
Not specified	
Xilinx Virtex-5	
AMD Virtex 5 FPGA ML510 embedded development platform	
Particle filter	MLM* [28]
Sum of Absolute Differences (SAD)	RM* [28]
Xilinx ML-507 evaluation board	
Mean shift algorithm	MLM [24]
Xilinx Virtex-7	
Xilinx VC709 evaluation board	
ILNET Convolutional neural networks (CNN)	DLM[20]

Table 8. Reported development platforms and algorithms for SoC-FPGA (n=11)

FPGA family **Board** Algorithm(s)	**Design methodology**
Intel Cyclone V	
Terasic DE1-SOC FPGA development board	
Adaptive Block Learning (FBS-ABL)	MLM* [35]
Binary Robust Invariant Scalable Keypoints (BRISK)	FM* [37]
Concurrent Zero Mean Normalized Cross-Correlation (ZNCC)	FM [32]
Fast Background Subtraction	RM* [35]
Harris corner detection	FM* [37]
Xilinx Artix-7	

ZC702 Evaluation Board for the Zynq-7000 XC7Z020 SoC	
Multidimensional Kalman filter	MLM [22]
Xilinx Zynq-7000	
AMD Zynq 7000 SoC ZC702 Evaluation Kit	
Attractor neural network (ANN)	DLM* [26]
Mean shift algorithm	MLM* [26]
AMD Zynq 7000 SoC ZC706 Evaluation Kit	
Honeybee search algorithm (HSA)	MLM* [33]
Simple elliptic matching	FM* [38]
Spatial 2-D DoG filtering	FM* [38]
Temporal high-pass filtering	FM* [38]
Zero Mean Normalized Cross-Correlation (ZNCC)	FM* [33]
No specified	
You Only Look Once (YOLO v3) CNN	DLM [30]
Xilinx Zedboard	
Background subtraction	RM* [34]
Discriminative scale space tracking (DSST) algorithm	MLM [19]
Independent component analysis (ICA)	MLM* [34]
Scalable particle filter	MLM [27]

Table 9. Reported development platforms and algorithms for MPSoC (n=3)

FPGA family **Board** Algorithm(s)	**Design methodology**
Xilinx Zynq Ultrascale+	
AMD Zynq UltraScale+ ZCU102 Evaluation Kit	
Efficent convolutional operators for tracking (ECO)	DLM* [23]
Kalman filter	MLM* [23]
AMD Zynq Ultrascale+ ZCU104 Evaluation kit	
Deep CNN single-shot multibox detector (SSD)	DLM* [39]
Kernelized correlation filters (KCF)	MLM* [39]
You Only Look Once (YOLO v3) CNN	DLM [29]

6 What Evaluation Techniques are Used for FPGA-based VOT Systems?

The effectiveness of FPGA-based VOT systems is influenced by various factors and design limitations. Evaluation of these systems require robust evaluation techniques to ensure accuracy, speed, and resource utilization under different circumstances. Moreover, incorporating VOT algorithms into FPGA-based platforms requires meticulous testing based on the selection of evaluation metrics to meet high performance requirements.

Assessing and comparison of tracking algorithms relies on three key elements: the selection or proposal of a dataset, the establishment of an evaluation protocol, and the use of performance evaluation metrics [42].

6.1 Selection or Proposal of a Dataset

Researchers of the VOT research field utilize a variety of datasets to evaluate the performance of their tracking algorithms (see Table 10). Dataset benchmarks are useful tools for defining standardized evaluation criteria and procedures to validate VOT systems under different conditions.

Other relevant datasets are toy datasets. They are a series of video sequences generated by system developers. They play a crucial role in FPGA-based accelerated computer vision research, offering tailored data that addresses specific requirements and challenges. By tailoring data to specific research objectives and providing detailed annotations, these datasets facilitate rigorous experimentation and evaluation, ultimately contributing to the development of robust and efficient vision systems for specific scenarios.

6.2 Establishment of an Evaluation Protocol.

Evaluation protocols for datasets often involve running tracking algorithms on the dataset sequences and computing the specified metrics using ground truth annotations. The most commonly utilized assessment techniques include one-pass evaluation (OPE), followed by the long-term oriented temporal resilience evaluation (TRE), and spatial resilience assessment (SRE) [22].

For the previous mentioned datasets, two OPE evaluation protocols were described in detail:
Kristan et al. [42] realized three experiments. For the first experiment, they tested all trackers across all sequences within the VOT2013 dataset. Initialization of the tracker was executed using the ground truth bounding boxes. The second experiment replicated experiment 1 but with a variation in initialization: they used perturbed bounding boxes instead of using precise ground truth bounding boxes. These perturbations were randomly generated, constituting approximately ten percent of the size of the ground truth bounding boxes. Finally, for the last experiment, the first experiment was repeated across all sequences, with a modification. Color images were converted to grayscale before conducting the tracking evaluation.

Table 10. Relevant dataset benchmarks for FPGA-based systems

Dataset	Year	Video sequences	Categories
ALOV++ [3]	2014	315	Light, Surface Cover, Specularity, Transparency, Shape, Motion Smoothness, Motion Coherence, Clutter, Confusion, Low Contrast, Occlusion, Moving Camera, Zooming Camera, Long Duration
Cdnet [43]	2014	53	Baseline, Camera Jitter, Dynamic background, Intermittent motion, Shadow, Thermal, Bad Weather, Low Framerate, Night video, PTZ, Turbulence
LaSOT [44]	2019	1,400	70 categories
OTB [1]	2015	100	Illumination Variation, Scale Variation, Occlusion, Deformation, Motion Blur, Fast Motion, In-Plane Rotation, Out-of-Plane, Out-of-View, Background Clutters, Low Resolution
VOT [42]	2013	60	Occlusion, Illumination change, Motion change, Size change, Camera motion

For the LaSOT dataset, Fan et al. [44] realized two experiments recommended for machine-learning VOT systems: In the first protocol, all 1,400 sequences are used for the assessment of tracking performance. The primary objective of this protocol is to facilitate a comprehensive evaluation of trackers on a large scale. The second protocol is designed to furnish a substantial collection of videos suitable for both training and evaluating trackers concurrently according to the Pareto principle.

It's important to mention that some datasets consider more than one of the mentioned assessment techniques. This is particularly relevant in the case of ALOV++ benchmark dataset, which includes the three assessment techniques: OPE, TRE and SRE [3].

6.3 Evaluation Metrics.

The challenge in evaluating visual tracking lies in the diversity of performance measures used by various researchers, leading to a lack of consensus on preferred metrics. As an effort to face this situation, Cehovin et al. [45] realize an in-depth study of VOT performance measures and propose as main recommended performance metrics the accuracy (using average overlap) and the robustness (using failure rate). At this point, Babu et al. [22] considers the assessment of tracking performance relies on precision and success rates as key

evaluation parameters, besides tracking speed (frame rate) in frames per second. On the other hand, El-Shafie and colleagues argue that establishing a distinct metric, such as OTB success AUC, across various designs is crucial for ensuring a fair comparison [20]. Other important evaluation metrics reported are recall, F-score, mean Average Precision (mAP), and area under curve (AUC).

Regarding FPGA-based platforms, the hardware resources consumption is reported as another comparison criteria. Resolution, Frame rate, Power consumption, Area resources usage (LUTs or ALUTs, LEs, DSPs, BRAM, FF's, I/O pins, etc.) are some related metrics for this topic.

By selecting and employing the right evaluation techniques, researchers and engineers can gain a comprehensive understanding of the performance and capabilities of FPGA-based VOT systems and identify areas for improvement.

6.4 Results and Discussion

For this study, a variety of evaluation techniques for FPGA-based VOT systems were reported, including toy datasets and dataset benchmarks, as well as evaluation metrics and performance measures. Notable findings are described below.

6.4.1 Datasets

For the 21 reviewed papers, the most reported dataset was OTB with 5 (24%) papers, featuring in several studies across different architectures and methodologies (see Tables 10, 11 and 12). CDnet and ALOV++ datasets appear with 2 (9.6%) and 1 (4.8%) papers respectively. Three papers (14.4%) reported toy datasets: in [36], a set of eight different images of 166 x 150 pixels was reported, in [26] sequences of more than one dataset ([22, 27, 23]), use of video camera sequences instead of dataset sequences ([24, 28]) or even more not reported datasets ([19, 31, 32]).

6.4.2 Evaluation Metrics

According to the findings of this study, although there are no standardized evaluation criteria for VOT systems, following interesting observations can be mentioned when FPGA-based platforms are considered (see Tables 11, 12 and 13). For the FPGA platforms, some authors prioritize real-time processing capabilities ([24, 31]) while others concentrate on image quality preservation and algorithm robustness ([21, 25]). In the case of MPSoC platforms, authors tend to focus on comprehensive performance evaluation encompassing precision and success rates ([23, 39]), essential for applications requiring high accuracy and reliability. Finally, SoC-FPGA platforms exhibit a wide range of evaluated metrics, from computation time to image quality metrics, reflecting the multifaceted nature of integrated systems and the need for tailored optimizations to meet application-specific requirements.

Table 11. Reported datasets for FPGA only platforms (n=7)

Datasets	Evaluated Metrics	Design methodology
Dataset generated by authors	Feature extraction algorithm robustness	RM* [36]
Dataset of the Weizmann Institute of Science	Structural similarity index metric (SSIM)	MLM [25]
Not specified	LBP algorithm detection time per frame	FM* [31]
OTB-100	Average computation time, OTB precision, OTB success AUC	DLM [20]
USC-SIPI image database	Maximum peak signal-to-noise ratio (PSNR)	DLM [21]
Video Camera used	Computation time for real-time object tracking	MLM [24]
Video Camera used	Not specified	RM* [28]

Table 12. Reported datasets for SoC-FPGA platforms (n=11)

Datasets	Evaluated Metrics	Design methodology

ALOV++	Average GPU-CPU seconds per frame, Average SoC-FPGA seconds per frame, F-score	FM* [33]
AVSS-2007, CAVIAR-2004, PETS-2006	Root mean square error (RMSE)	MLM [27]
CDnet 2014	Average precision, Average recall, Total runtime per frame	RM* [34]
CDnet 2014	Average precision, Average recall, F-score	RM* [35]
Dataset generated by authors	Average computation time, Average tracking performance	FM* [37]
Dataset generated by authors	Computation time for Mean-shift method, Computation time for Particle filter method	MLM* [26]
MOT-16, OTB-100, UAVDT	Precision, Success rate (AUC)	MLM [22]
Not specified	Not specified	FM [32]
Not specified	Time results for SVD implementation in FPGA	MLM [19]
OTB-100	Average precision	FM* [38]
UADETRAC (modified)	Mean average precision (mAP)	DLM [30]

Table 13. Reported datasets for MPSoC platforms (n=3)

Datasets	Evaluated Metrics	Design methodology
Cityscape dataset	Not specified	DLM [29]
LaSOT, OTB-100	AUC score, Mean average precision (mAP)	MLM* [23]
OTB-100	Precision rate (PR), Success rate (SR)	MLM* [39]

6.4.3 Performance Measures

Table 14 shows the analysis of performance measures across FPGA, MPSoC, and SoC-FPGA platforms for the reviewed papers of this work. The selected performance measures show an elemental comparison between reported VOT systems: tracking speed (in frames per second, fps), image resolution (in pixels) and power consumption (in Watts, W). Notably, despite its relevance, eleven papers do not report one or more of these measures. These FPGA implementations exhibit diverse tracking speeds and resolutions, ranging from 13.42 fps to 8771 fps and from 300x420 to 1920x1080, respectively. Regarding the reported maxima measure of 8771 fps, this paper ([25]) uses a parallelization with an approximate computing technique to reach this performance. Finally, for power consumption measure, although this is the measure less reported with just 12 cases, its reported range goes from 24mW to 12.51W, indicating a variance in the focus on power efficiency across different research endeavors.

Table 14. Performance measures (n=21)

Platform	Design metho-dology	Authors	Tracking speed (frames per second, fps)	Max. reported resolution (width)	Max. reported resolution (height)	Power consumption (W)
FPGA only	RM*	[36]	44	1920	1080	Not specified
	RM*	[28]	303	720	576	Not specified
	FM*	[31]	30	1024	768	Not specified
	MLM	[24]	60	640	480	3.15
	MLM	[25]	8771	300	420	Not specified
	DLM	[20]	44	Not specified	Not specified	Not specified
	DLM	[21]	434	640	480	Not specified
MPSoC	MLM*	[23]	13.42	1920	1080	4
	MLM*	[39]	36.2	Not specified	Not specified	Not specified
	DLM	[29]	27	320	240	8.39
SoC-FPGA	RM*	[34]	Not specified	320	200	4.2

RM*	[35]	83	320	240	Not specified
FM	[32]	Not specified	640	480	Not specified
FM*	[38]	600	320	240	0.768
FM*	[37]	60	640	480	0.024
FM*	[33]	27	480	360	5
MLM	[22]	91	Not specified	Not specified	0.78
MLM	[27]	348	320	240	0.6
MLM	[19]	25.38	320	240	1.92
MLM*	[26]	30	1024	768	0.03
DLM	[30]	91.65	416	416	12.51

7 Conclusions and Perspectives

In recent years, there has been a rise in the prominence of FPGA technologies, showing promising potential and capabilities for implementing a variety of systems, from autonomous systems to edge computing applications ([12, 46]). In this review, an analysis of the 21 selected articles regarding design methodologies, hardware implementations and evaluation techniques was performed, in order to propose an insight of the major perspectives of the FPGA-based VOT systems. According to our research, replication of some approaches may be difficult due to the lack of detailed information.

The analysis focused on algorithm-based design methodologies from the selected papers reveals a significant trend towards hybrid systems, particularly in utilizing two or more algorithms to overcome limitations inherent non-hybrid methodologies In particular, most of the reported algorithms belong to the MLM categories, closely followed by the FM and DLM categories, with the RM and FM algorithms being mostly part of hybrid systems. In particular, a variety of algorithms for VOT were reported, including Kalman filter, Mean shift, and YOLO-based CNNs with offline training. It is worth mentioning that selecting algorithms based on object characteristics would be a challenging design and implementation task, due to the computational resources available in FPGA platforms.

Regarding the implementation of high-performance FPGA-based VOT systems, the choice of FPGA-based platform would heavily depend on the specific requirements of the application, considering factors such as real-time performance, precision, computational efficiency, and power constraints. HW/SW co-design approach is used to optimize the performance of these systems, although this is not detailed in several works. VHDL is commonly used for Intel boards and a mix of VHDL, HLS, C, RTL, and Python for Xilinx boards. In addition, MATLAB is frequently utilized for numerical simulation, algorithm evaluation, and transition from PC-based to FPGA implementation.

FPGA components from different platforms are mainly used for computing-intensive tasks, such as pixel-level operations and high data transfer between different architectures. Moreover, optimization techniques such as pipelining and parallelism are widely employed to enhance speed and efficiency.
Evaluation techniques are crucial in the VOT systems development. OTB dataset was the most commonly used, followed by CDnet and ALOV++ datasets. Some papers reported datasets generated by the authors (toy datasets). Relevant metrics reported hierarchically are precision, recall, frames per second, among others. Regarding FPGA performance measures such as speed, image resolution, and power consumption are reported.

Overall, this review generates the following recommendations and perspectives to provide an overview in different aspects to the community interested in the development of FPGA-based VOT systems.

Careful selection of algorithms based on criteria such as characteristics of the objects being tracked, available computational resources, and environmental conditions is recommended to address the complexity of VOT problems. It is important to leverage the complementary strengths of various algorithms as part of a hybrid methodologies to achieve superior performance and adaptability in real-world scenarios.

Generally speaking, the SoC-FPGA platforms are more suitable for developing FPGA-based VOT system proposals compared to FPGA and MPSoC platforms, due to their HW flexibility and scalability, software

development alternatives at different levels of abstraction or approaches, technical support, and affordable cost. However, it is important to mention that limited computing resources, power consumption, learning curve, and financial constraints might present issues in VOT implementation. For perspective, an in-depth analysis of different FPGA-based VOT systems will provide additional insights into the advantages and disadvantages of using different FPGA-based platforms.

Dataset benchmarks and toy datasets have specific evaluation purposes at different stages of VOT development. Nevertheless, dataset benchmarks provide additional efficient tools for VOT system validation against relevant state-of-the-art trackers. Therefore, the recommendation is to select specific video sequences included in dataset benchmarks to evaluate the performance of novel FPGA-based VOT systems considering the constraints of custom design. In perspective, future research should focus on emphasizing the standardization of datasets, metrics and measurements to refine the evaluation techniques of FPGA-based VOT systems.

8 Acknowledgements

This research was supported by a CONACYT Grant scholarship: 2022-2026, for the doctoral studies of the first author, and was funded by CONACYT through the grant "Convocatoria de Ciencia Básica y/o Ciencia de Frontera 2022", project ID 320036.

References

1. Wu, Y., Lim, J., & Yang, M.-H. (2013). Online object tracking: A benchmark. En *2013 IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2411–2418). Portland, OR, USA: IEEE. https://doi.org/10.1109/CVPR.2013.312
2. El-Shafie, A.-H. A., & Habib, S. E. D. (2019). Survey on hardware implementations of visual object trackers. *IET Image Processing, 13*(6), 863–876. https://doi.org/10.1049/iet-ipr.2018.5952
3. Smeulders, A. W. M., Chu, D. M., Cucchiara, R., Calderara, S., Dehghan, A., & Shah, M. (2014). Visual tracking: An experimental survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 36*(7), 1442–1468. https://doi.org/10.1109/TPAMI.2013.230
4. Bhowmik, D., & Appiah, K. (2018). Embedded vision systems: A review of the literature. En *Applied Reconfigurable Computing. Architectures, Tools, and Applications* (pp. 204–216). Cham: Springer International Publishing. https://doi.org/10.1007/978-3-319-78890-6_17
5. Molina, R. S., Gil-Costa, V., Crespo, M. L., & Ramponi, G. (2022). High-level synthesis hardware design for FPGA-based accelerators: Models, methodologies, and frameworks. *IEEE Access, 10*, 90429–90455. https://doi.org/10.1109/ACCESS.2022.3201107
6. Li, M., Cai, Z., Wei, C., & Yuan, Y. (2015). A survey of video object tracking. *International Journal of Control and Automation, 8*(9), 303–312. https://doi.org/10.14257/ijca.2015.8.9.29
7. Fiaz, M., Mahmood, A., Javed, S., & Jung, S. K. (2019). Handcrafted and deep trackers: Recent visual object tracking approaches and trends. *arXiv*. https://doi.org/10.48550/arXiv.1812.07368
8. Zhang, Y., Wang, T., Liu, K., Zhang, B., & Chen, L. (2021). Recent advances of single-object tracking methods: A brief survey. *Neurocomputing, 455*, 1–11. https://doi.org/10.1016/j.neucom.2021.05.011
9. Tiwari, M. (2017). A review of detection and tracking of object from image and video sequences. Recuperado de https://www.semanticscholar.org/paper/A-Review-of-Detection-and-Tracking-of-Object-from-Tiwari/b746d35a4b8b4f57dcc030043dd329ff79285b19 [Accedido el 3 de julio de 2023].
10. Zeng, K., Ma, Q., Wu, J. W., Chen, Z., Shen, T., & Yan, C. (2022). FPGA-based accelerator for object detection: A comprehensive survey. *Journal of Supercomputing, 78*(12), 14096–14136. https://doi.org/10.1007/s11227-022-04415-5
11. Feng, X., Jiang, Y., Yang, X., Du, M., & Li, X. (2019). Computer vision algorithms and hardware implementations: A survey. *Integration, 69*, 309–320. https://doi.org/10.1016/j.vlsi.2019.07.005
12. Seng, K. P., Lee, P. J., & Ang, L. M. (2021). Embedded intelligence on FPGA: Survey, applications, and challenges. *Electronics, 10*(8). https://doi.org/10.3390/electronics10080895
13. Saha, S., & Bhattacharyya, S. S. (2009). Design methodology for embedded computer vision systems. En *Embedded Computer Vision* (pp. 27–47). London: Springer. https://doi.org/10.1007/978-1-84800-304-0_2
14. Ali, A., et al. (2016). Visual object tracking—Classical and contemporary approaches. *Frontiers of Computer Science, 10*(1), 167–188. https://doi.org/10.1007/s11704-015-4246-3
15. Salau, A. O., & Jain, S. (2019). Feature extraction: A survey of the types, techniques, applications. En *2019 International Conference on Signal Processing and Communication (ICSC)* (pp. 158–164). https://doi.org/10.1109/ICSC45622.2019.8938371

16. Karimi-Mamaghan, M., Mohammadi, M., Meyer, P., Karimi-Mamaghan, A. M., & Talbi, E.-G. (2022). Machine learning at the service of meta-heuristics for solving combinatorial optimization problems: A state-of-the-art. *European Journal of Operational Research*, *296*(2), 393–422. https://doi.org/10.1016/j.ejor.2021.04.032
17. Sarker, I. H. (2021). Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, *2*(6), 420. https://doi.org/10.1007/s42979-021-00815-1
18. Bailer, C., & Stricker, D. (2015). Tracker fusion on VOT challenge: How does it perform and what can we learn about single trackers? En *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)* (pp. 630–638). https://doi.org/10.1109/ICCVW.2015.85
19. Walid, W., Awais, M., Ahmed, A., Masera, G., & Martina, M. (2021). Real-time implementation of fast discriminative scale space tracking algorithm. *Journal of Real-Time Image Processing*, *18*(6), 2347–2360. https://doi.org/10.1007/s11554-021-01119-6
20. El-Shafie, A.-H. A., Zaki, M., & Habib, S. E. D. (2022). An efficient hardware implementation of CNN-based object trackers for real-time applications. *Neural Computing and Applications*, *34*(22), 19937–19952. https://doi.org/10.1007/s00521-022-07538-1
21. Huang, Z., et al. (2017). A vector-quantization compression circuit with on-chip learning ability for high-speed image sensor. *IEEE Access*, *5*, 22132–22143. https://doi.org/10.1109/ACCESS.2017.2762399
22. Babu, P., & Parthasarathy, E. (2022). FPGA implementation of multi-dimensional Kalman filter for object tracking and motion detection. *Engineering Science and Technology, an International Journal*, *33*, 101084. https://doi.org/10.1016/j.jestch.2021.101084
23. Iqbal, O., Muro, V. I. T., Katoch, S., Spanias, A., & Jayasuriya, S. (2022). Adaptive subsampling for ROI-based visual tracking: Algorithms and FPGA implementation. *IEEE Access*, *10*, 90507–90522. https://doi.org/10.1109/ACCESS.2022.3200755
24. Pandey, J. G. (2021). An embedded FPGA-SoC framework and its usage in moving object tracking application. *Design Automation for Embedded Systems*, *25*(3), 213–236. https://doi.org/10.1007/s10617-021-09252-y
25. Tehreem, A., Khawaja, S. G., Khan, A. M., Akram, M. U., & Khan, S. A. (2019). Multiprocessor architecture for real-time applications using mean shift clustering. *Journal of Real-Time Image Processing*, *16*(6), 2233–2246. https://doi.org/10.1007/s11554-017-0733-0
26. Yang, S., Wong-Lin, K., Andrew, J., Mak, T., & McGinnity, T. M. (2018). A neuro-inspired visual tracking method based on programmable system-on-chip platform. *Neural Computing and Applications*, *30*(9), 2697–2708. https://doi.org/10.1007/s00521-017-2847-5
27. Engineer, P., Velmurugan, R., & Patkar, S. (2020). Scalable implementation of particle filter-based visual object tracking on network-on-chip (NoC). *Journal of Real-Time Image Processing*, *17*(5), 1117–1134. https://doi.org/10.1007/s11554-018-0841-5
28. Singh, S., Shekhar, C., & Vohra, A. (2017). Real-time FPGA-based object tracker with automatic pan-tilt features for smart video surveillance systems. *Journal of Imaging*, *3*(2). https://doi.org/10.3390/jimaging3020018
29. Cittadini, E., Marinoni, M., Biondi, A., Cicero, G., & Buttazzo, G. (2023). Supporting AI-powered real-time cyber-physical systems on heterogeneous platforms via hypervisor technology. *Real-Time Systems*, *59*(4), 609–635. https://doi.org/10.1007/s11241-023-09402-4
30. Zhai, J., Li, B., Lv, S., & Zhou, Q. (2023). FPGA-based vehicle detection and tracking accelerator. *Sensors*, *23*(4). https://doi.org/10.3390/s23042208
31. Wu, W., Su, D., Yuan, B., & Li, Y. (2021). Intelligent security monitoring system based on RISC-V SoC. *Electronics*, *10*(11), 1366. https://doi.org/10.3390/electronics10111366
32. Chen, Z., Li, S., Zhang, N., Hao, Y., & Zhang, X. (2019). Eye-to-hand robotic visual tracking based on template matching on FPGAs. *IEEE Access*, *7*, 88870–88880. https://doi.org/10.1109/ACCESS.2019.2926807
33. Soubervielle-Montalvo, C., et al. (2022). Design of a low-power embedded system based on a SoC-FPGA and the Honeybee search algorithm for real-time video tracking. *Sensors*, *22*(3), 1280. https://doi.org/10.3390/s22031280
34. Carrizosa-Corral, F., et al. (2018). FPGA-SoC implementation of an ICA-based background subtraction method. *International Journal of Circuit Theory and Applications*, *46*(9), 1703–1722. https://doi.org/10.1002/cta.2544
35. Montero, V. J., Jung, W.-Y., & Jeong, Y.-J. (2021). Fast background subtraction with adaptive block learning using expectation value suitable for real-time moving object detection. *Journal of Real-Time Image Processing*, *18*(3), 967–981. https://doi.org/10.1007/s11554-020-01058-8
36. Aguilar-González, A., Arias-Estrada, M., & Berry, F. (2018). Robust feature extraction algorithm suitable for real-time embedded applications. *Journal of Real-Time Image Processing*, *14*(3), 647–665. https://doi.org/10.1007/s11554-017-0701-8
37. Nam, T., Kim, S., & Jung, D. (2019). Hardware implementation of KLT tracker for real-time intruder detection and tracking using on-board camera. *International Journal of Aeronautical and Space Sciences*, *20*(1), 300–314. https://doi.org/10.1007/s42405-018-0131-2
38. He, W., et al. (2021). A low-cost high-speed object tracking VLSI system based on unified textural and dynamic compressive features. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *68*(3), 1013–1017. https://doi.org/10.1109/TCSII.2020.3020883

39. Ji, Q., Dai, C., Hou, C., & Li, X. (2021). Real-time embedded object detection and tracking system in Zynq SoC. *Journal of Image and Video Processing*, *2021*(1), 21. https://doi.org/10.1186/s13640-021-00561-7
40. Ha, S., & Teich, J. (2017). Introduction to hardware/software codesign. En *Handbook of Hardware/Software Codesign* (pp. 3–26). Dordrecht: Springer. https://doi.org/10.1007/978-94-017-7267-9_41
41. Ahmad, A., Al Busaidi, S. S., Al-Maashri, A., Awadallah, M., & Hussain, S. (2021). FPGAs – Chronological developments and challenges. *International Journal of Electrical Engineering and Technology*, *12*(11).
42. Kristan, M., Pflugfelder, R., Leonardis, A., Matas, J., & Porikli, F. (2013). The Visual Object Tracking VOT2013 challenge results.
43. Wang, Y., Jodoin, P.-M., Porikli, F., Konrad, J., Benezeth, Y., & Ishwar, P. (2014). CDnet 2014: An expanded change detection benchmark dataset. En *2014 IEEE Conference on Computer Vision and Pattern Recognition Workshops* (pp. 393–400). Columbus, OH, USA: IEEE. https://doi.org/10.1109/CVPRW.2014.126
44. Fan, H., et al. (2019). LaSOT: A high-quality benchmark for large-scale single object tracking. En *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 5369–5378). Long Beach, CA, USA: IEEE. https://doi.org/10.1109/CVPR.2019.00552
45. Čehovin, L., Kristan, M., & Leonardis, A. (2014). Is my new tracker really better than yours? En *IEEE Winter Conference on Applications of Computer Vision* (pp. 540–547). https://doi.org/10.1109/WACV.2014.6836055
46. Xu, C., et al. (2022). The case for FPGA-based edge computing. *IEEE Transactions on Mobile Computing*, *21*(7), 2610–2619. https://doi.org/10.1109/TMC.2020.3041781