Computational Imaging for Energy-Efficient Cameras:

Adaptive ROI-based Object Tracking and Optically Defocused Event-based Sensing.

by

Victor Isaac Torres Muro

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2022 by the
Graduate Supervisory Committee:

Suren Jayasuriya, Chair
Andreas Spanias
Jae-sun Seo

ARIZONA STATE UNIVERSITY

May 2022

ABSTRACT

Computer vision is becoming an essential component of embedded system applications such as smartphones, wearables, autonomous systems and internet-of-things (IoT). These applications are generally deployed into environments with limited energy, memory bandwidth and computational resources. This trend is driving the development of energy-efficient image processing solutions from sensing to computation. In this thesis, different alternatives are explored to implement energy-efficient computer vision systems.

First, I present a field programmable gate array (FPGA) implementation of an adaptive subsampling algorithm for region-of-interest (ROI) -based object tracking. By implementing the computationally intensive sections of this algorithm on an FPGA, I aim to offload computing resources from energy-inefficient graphics processing units (GPUs) and/or general-purpose central processing units (CPUs). I also present a working system executing this algorithm in near real-time latency implemented on a standalone embedded device.

Secondly, I present a neural network-based pipeline to improve the performance of event-based cameras in non-ideal optical conditions. Event-based cameras or dynamic vision sensors (DVS) are bio-inspired sensors that measure logarithmic per-pixel brightness changes in a scene. Their advantages include high dynamic range, low latency and ultra-low power when compared to standard frame-based cameras. Several tasks have been proposed to take advantage of these novel sensors but they rely on perfectly calibrated optical lenses that are in-focus. In this work I propose a method to reconstruct events captured with an out-of-focus event-camera so they can be fed into an intensity reconstruction task. The network is trained with a dataset generated by simulating defocus blur in sequences from object tracking datasets such as LaSOT and OTB100. I also test the generalization performance of this network in scenes captured with a DAVIS event-based sensor equipped with an out-of-focus lens.

DEDICATION

To my parents.

ACKNOWLEDGMENTS

I would first like to extend sincere thanks to my advisor Dr. Suren Jayasuriya for his constant support and encouragement in the pursuit of my academic degree and research goals. His advice and motivation have been critical in my development as a student and scholar. I am filled with gratitude for the opportunity he gave me to work in his computer vision lab.

I owe a great debt of gratitude to Odrika Iqbal for her leadership and guidance on our Adaptive Subsampling journal paper. Chapter 3 is a revision of the work we did for over a year. She designed the algorithm and ran the software experiments (AUC scores, keyframing, memoization and CMOS power consumption) that paved the way for the hardware implementation on an FPGA.

I would also like to thank Joshua Rego for his introduction to lensless imaging systems and hardware experiments. To him I owe the point-spread-function and data captured with an event camera that I used in my experiments.

Additionally, I wish to thank the Astrobotic team: Andrew Horchler, Chris Owens, Michael Bloom and Colman Glagovich for the knowledge and discussion shared about event-based cameras and FPGAs.

I would like to thank all the members of the Imaging Lyceum team including Albert Reed, Gregory Vetaw, Ripon Kumar, Olivia Christie and Ali Almuallem for their ecouragement, constructive conversations and friendship.

The US-Mexico Commission for Educational & Cultural Exchange (Fulbright-Comexus) and the Graduate College generously funded me through the course of my graduate degree at Arizona State University, and I am immensely grateful for the opportunities they offered me that allowed me to achieve my dream.

Finally I would like to thank my parents whose unconditional love, inspiration and support enabled me to be the person I am today. From them I've learned the values of honesty and hard work. They have always supported my sister, brother and me in whatever academic undertaking we chose to pursue. I am forever indebted to them. This thesis is dedicated to them.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Figure Page

Chapter 1

INTRODUCTION

The field of computer vision has attained significant advances in recent years. With the help of machine learning tools, several state-of-the-art algorithms have been developed in order to aid computational systems to make decisions relying only on visual data. At the same time, strong emphasis is being placed on the deployment of these algorithms in day-to-day settings such as mobile systems and Internet of Things (IoT) applications.

Image sensing is the first stage of any computer vision application. And although most modern mobile systems employ CMOS image sensors due to their low power and low cost, it is power hungry. Hundreds of milliWatts are consumed due to narrowly defined hardware and software interfaces and even reducing the image quality does not provide significant power reduction [44].

After sensing, computer vision algorithms rely on general-purpose computational devices such as multicore CPUs or graphic processing units (GPUs) to execute image processing tasks. Extensive work has shown that these devices are not the most energy-efficient solution for complex computer tasks [62]. Given the battery and computational constraints of such devices, there is a strong focus on the pursuit of energy-efficient computational techniques and low-power sensing to achieve the deployment of modern computer vision innovations under these limitations.

The central idea of this thesis is to present different approaches for low power computer vision. The first is based on implementing an adaptive subsampling object tracking algorithm on a field programmable gate array (FPGA). Adaptive subsampling can reduce power consumption by leveraginga camera's of region-of-interest (ROI). The ROI is defined by a bounding box enclosing the area of an image frame in which the target object exists. ROI-based energy optimization entails the discarding of pixels outside of the ROI and selectively reading out only the pixels comprising the moving object resulting in faster bandwidth and improved energy efficiency [73, 87].

As mentioned, we implement this algorithm on an FPGA given that they are becoming a popular energy-efficient embedded platform for IoT applications when used at the edge computing paradigm. This new paradigm aims to substitute a portion of the computation typically performed in remote, cloud-based, energy-demanding servers by performing computation near the end device. Typically, these servers rely on CPUs and GPUs to process batches of time-insensitive data in a centralized environment. Using FPGAs at the edge aids in the reduction of power consumption, increased energy efficiency and better thermal stability compared to general purpose devices [8]. They offer opportunities to improve algorithm performance by exploiting low-level fine-grained parallelism by customizing data paths to the requirements of a specific application [62].

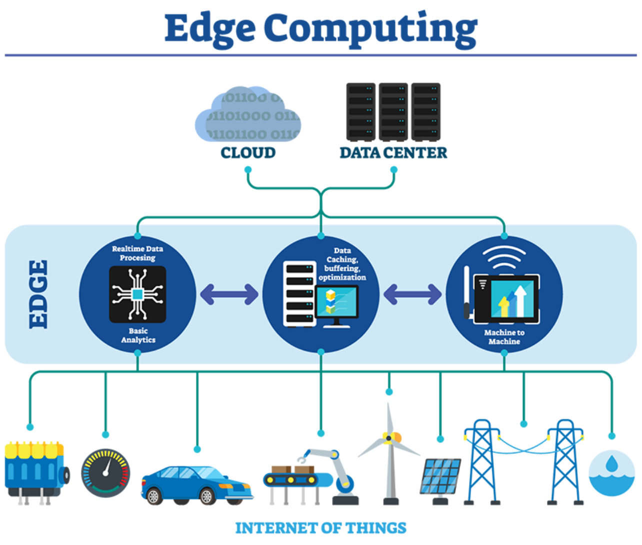


Figure 1. Edge computing paradigm.[63]

Our second approach takes advantage of a low-power sensors known as event-based cameras. These sensors change the paradigm of visual information by capturing changes in logarithmic light intensity at an asynchronous rate instead of synchronously storing frames, even if there are no changes in the scene. Apart from the low-power sensing, these sensors provide other advantages such as high dynamic range, low data bandwidth consumption and high temporal resolution [22].

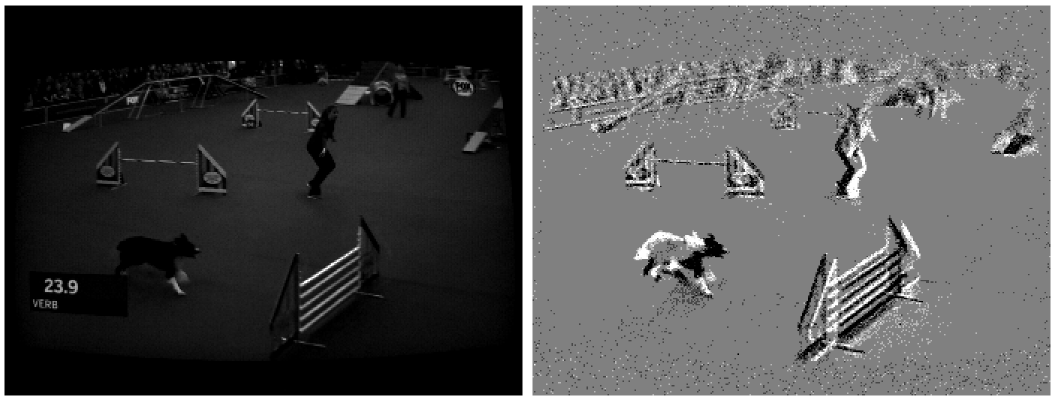


Figure 2. Comparison of the output from an standard camera (left) and an event-based camera (right).

Given the nature of their asynchronous output, a pre-processing step is required to reconstruct events into intensity information before modern computer vision algorithms can be used. There are several intensity reconstruction algorithm from event data in the literature [67][75][10]. However, they rely on near-perfect optical focus in order to reconstruct high-quality frames. Whenever optical imperfections such as blur from an out-of-focus lens occur, these algorithms are not able to successfully reconstruct intensity information from events. We aim to improve the robustness of these dynamic vision sensors by adding a neural network that reconstructs out-of-focus event voxels into high-quality voxels to improve intensity reconstruction.

This work paves the way to reconstruct out-of-focus events that can be used in a multiple variety of tasks such as optical flow and object detection, not only intensity reconstruction. Also, given that reconstruction from out-of-focus information is essentially a point-spread function (PSF) deconvolution problem, we aim to make our approach applicable to other PSF deconvolution problems such as scenes captured with a lensless event-based camera.

1.1 Contributions

The main contributions in this thesis are the following:

- FPGA implementation of adaptive subsampling for ROI-based object tracking taking advantage of state-of-the-art high-level synthesis (HLS) tools.
- A robust intensity reconstruction pipeline from events captured by an out-of-focus event-based camera.

1.2 Outline

In Chapter 2, an introductory background on concepts related to this thesis is presented. In Chapter 3, we introduce the FPGA implementation of an adaptive subsampling pipeline for ROI-based object tracking algorithm. We talk about the algorithm development and the high-level synthesis (HLS) tools employed to implement this algorithm on an embedded device. In Chapter 4, we present a machine learning-based pipeline employed on the reconstruction of scenes captured from an out-of-focus event camera. Chapter 5 concludes the work in this thesis. Each chapter consists of a background, approach, implementation, and results section; presented in chronological order of when the work was performed.

Chapter 2

BACKGROUND

This chapter introduces the background topics that can enable embedded computer vision. This section will initially cover the subjects of embedded computer vision, software-defined imaging and region of interest (ROI) to help the reader familiarize with the concepts discussed through this thesis. Afterwards, we will delve into the two specific hardware platforms we focused on for this work. The first are the re-configurable computational devices known as Field Programmable Gate Arrays (FPGAs). Finally, we will introduce the function and applications of event-based cameras or dynamic vision sensors.

2.1 Embedded computer vision.

Computer vision is becoming an essential component of embedded system applications such as smartphones, autonomous systems and internet-of-things. Currently, these systems rely on traditional computer vision techniques which follow compute intensive brute-force approaches. This trend is driving a development of energy-efficient image processing solutions that can be deployed on environments with limited power, bandwidth and computational resources.

Discussion of vision based systems and approaches, and how they have been implemented on embedded devices is covered in [7]. This work also covers advantages and disadvantages of various embedded implementations and an overview of the challenges in the field as well as future research trends.

In [62], the authors evaluate the different approaches that industry and academia have explored in order to accelerate computer vision tasks in embedded applications. They compare multi-core central processing units (CPUs), graphic processing units (GPUs) and FPGAs, evaluating their performance characteristics depending on the complexity of the task.

2.2 Software-defined imaging.

Huge advancements have been made over the years in terms of image sensing hardware and vision data processing algorithms. However, there exists a gap between hardware and software design in an imaging system. Bridging this gap is key to achieving improved energy efficiency as well as latency.

In [33] the authors explore the existing works in the vision literature which can be leveraged to replace conventional hardware components in an imaging system with software such that it enables reconfigurability. They refer to this as software-defined imaging (SDI), where image sensor behaviour can be altered by the system software depending on the user needs. SDI centers around the idea of the traditional hardware components and mechanisms of image sensors being replaced or augmented by software alternatives.

2.3 Region of interest.

One method of achieving favorable tradeoffs between energy-efficiency and task performance is to only read the set of pixels within an image frame which describe the target object. The term region of interest or ROI can be considered as any continuous set of pixels that can be read from an image sensor.

Since the development of the ROI, many ROI-based solutions have generated a lot of interest from the early 2000s. These applications include ROI-based object tracking and detection for computer vision and foveated imaging. An integration of these algorithm with ROI camera sensors implies tremendous power savings for both the camera readout as well as the downstream image processing due to data reduction and latency improvements. Works such as [28, 46] show the early advances of ROI extraction during this years. Early ROI selection algorithms also utilized the Kalman filter, which adapts its internal matrices and state vector according to external measurements [9, 50, 41, 36]. More recently, Zhang *et al.* showed how to use frequency and space domain features for localizing regions of interest [97]. With growing interest in neural network solutions, several deep learning-based object detection/tracking methods have been developed which offer state-of-the-art ROI-ing capability [86, 55, 18, 79].

For energy-efficient region-of-interest (ROI) algorithms, recent research has introduced the concept of adaptive subsampling where image sensors can powergate or turn off pixels outside of the ROI during readout to save power. Adaptive subsampling works by defining and ROI that localizes part of the frame the user is interested in, and the pixels outside of the ROI are deemed redundant and discarded [53]. Recent work leveraged a YOLO detector and Kalman filter coupled together for predictive ROI tracking [31] in order to facilitate adaptive subsampling, the end goal being energy optimization via preemptive pixel inactivation outside of the region of interest. The work demonstrated how adaptive image sampling has tremendous potential in reducing computational complexity and processing speed.

2.4 Field Programmable Gate Arrays

Field Programmable Gate Arrays (FPGAs) are semiconductor devices that are based around a matrix of configurable logic blocks (CLBs) connected via programmable interconnects [88].

Compared to Application Specific Integrated Circuits (ASICs), FPGAs main advantage is that their hardware resources are re-programmable. This means that FPGAs can change the majority of the electrical functionality inside the device, even after it is manufactured, assembled or shipped. This flexibility allows a single FPGA device to be used for a wide variety of applications which in turn results in a reduced time to market when compared to ASICs. [89].

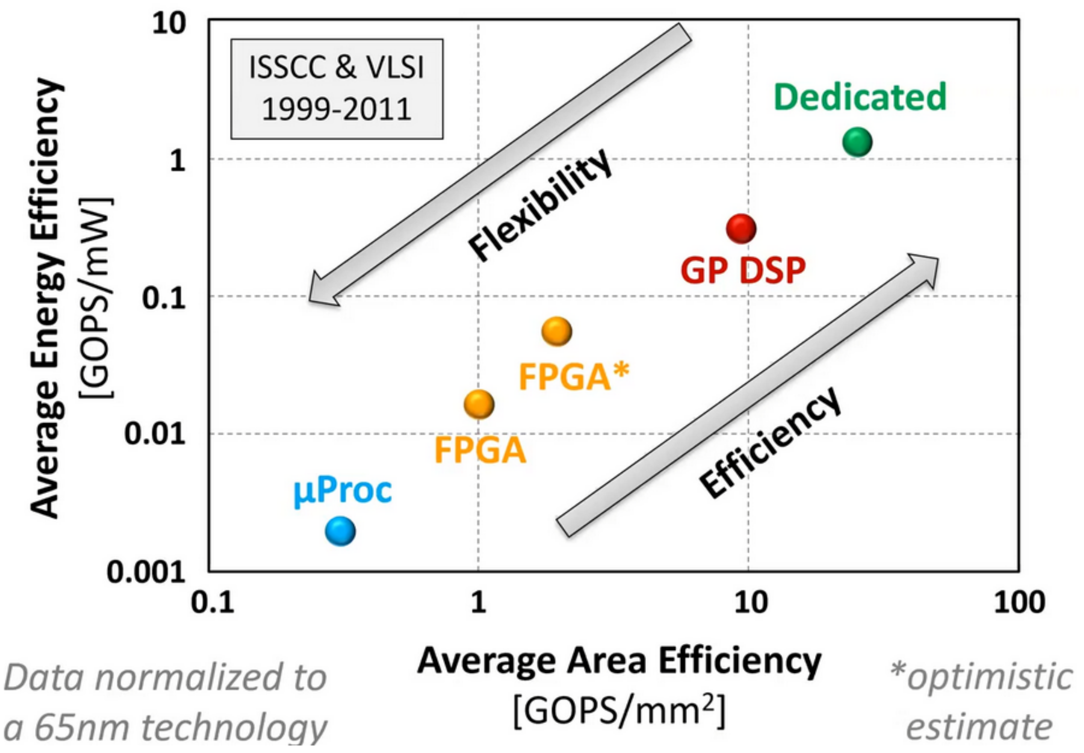


Figure 3. Efficiency and flexibility comparison of diverse hardware platforms. [70]

Another advantage of FPGAs is that by re-configuring into a specific hardware architecture, algorithms can be offloaded from general-purpose computing devices. By taking advantage of temporal and spatial parallelism, this "acceleration" allows the entire system to improve its performance and energy efficiency when compared to CPUs or GPUs.

FPGAs are also becoming popular on the edge-computing paradigm for Internet of Things (IoT) applications. They provide consistent throughput invariant to the size of application work-load, spatial and temporal parallelism and 3-4 times lower power consumption and up to 30.7 times better energy efficiency when compared to GPUs [8]. This makes them an ideal candidate to offload computation from centralized and energy-demanding cloud servers.

In order to take full advantage of FPGAs reprogrammability, the user requires proficiency in Hardware Description Languages (HDL). These programming languages are used to describe the structure and behaviour of digital logic circuits. This knowledge however, represented an enormous barrier for algorithm developers in order to accelerate algorithms using FPGA technology. Recently, High-Level Synthesis tools (HLS) have been developed in order to bridge the gap between software and hardware. These tools allow the developer to synthesize accelerated applications that can run on FPGAs without the knowledge of any specific HDL. Several manufacturers now provide HLS tools for their FPGA families [56] that have allowed algorithm developers to take advantage of this technology.

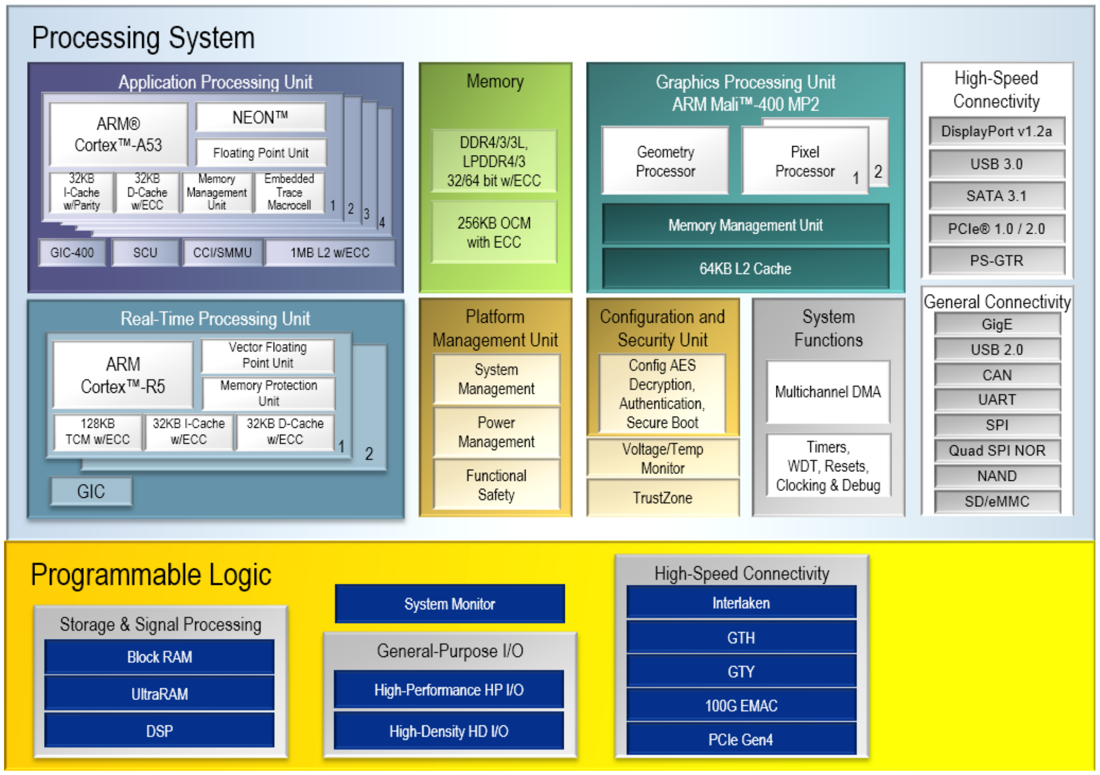


Figure 4. Zynq UltraScale+ EG MPSoC ZCU102 Block Diagram: The processing system contains an ARM Cortex A53 CPU along with DDR4 memory and several high-speed connectivity standards. The programmable logic contains the configurable memory, signal processing, connectivity and I/O resources [106].

2.5 Event Cameras

The past few decades of computer vision research have been devoted predominantly to standard frame-based cameras. However, the information captured by traditional cameras is designed for the use of humans, not machines. Images are synchronously generated with fixed exposure times and measure the absolute intensity of light in a scene. Given these properties they suffer from a limited dynamic range, meaning that they lack the ability to capture information in extreme lighting conditions, and they are also susceptible to motion blur on highly dynamic scenes. Another drawback of conventional cameras is that they capture information regardless of the scene properties. Even if there are no changes in the scene, frame-based cameras capture redundant information that consumes power and bandwidth.

Event-based cameras are novel vision sensors that aim to improve upon standard imaging systems. Also called Dynamic Vision Sensors (DVS) or Silicon Retinas, these sensors are asynchronous, bio-inspired and measure per-pixel logarithmic brightness changes in a scene. Instead of frames, an event camera outputs a stream of events that encodes time, location and polarity of the logarithmic intensity change at a certain pixel.

An event e_i on pixel (x_i, y_i):

$$e_i = \{x_i, y_i, t_i, p_i\} \tag{2.1}$$

is fired when the logarithmic change from time t_{i-1} and t_i is above the threshold θ:

$$|log(I_{t_i}(x, y) - log(I_{t_{i-1}}(x, y)| > \theta \tag{2.2}$$

In contrast with the conventional frame-based cameras, event cameras have superior properties such as very high temporal resolution and low latency (in the order of microseconds), very high dynamic range (140 dB vs 60 dB of standard cameras), and low power consumption (1mW vs 1W)[22][43]. Hence, event cameras have large potential for computer vision applications that demand low-power consumption, such as robotics and wearable devices or in challenging scenarios for standard cameras such as high speed and high dynamic range.

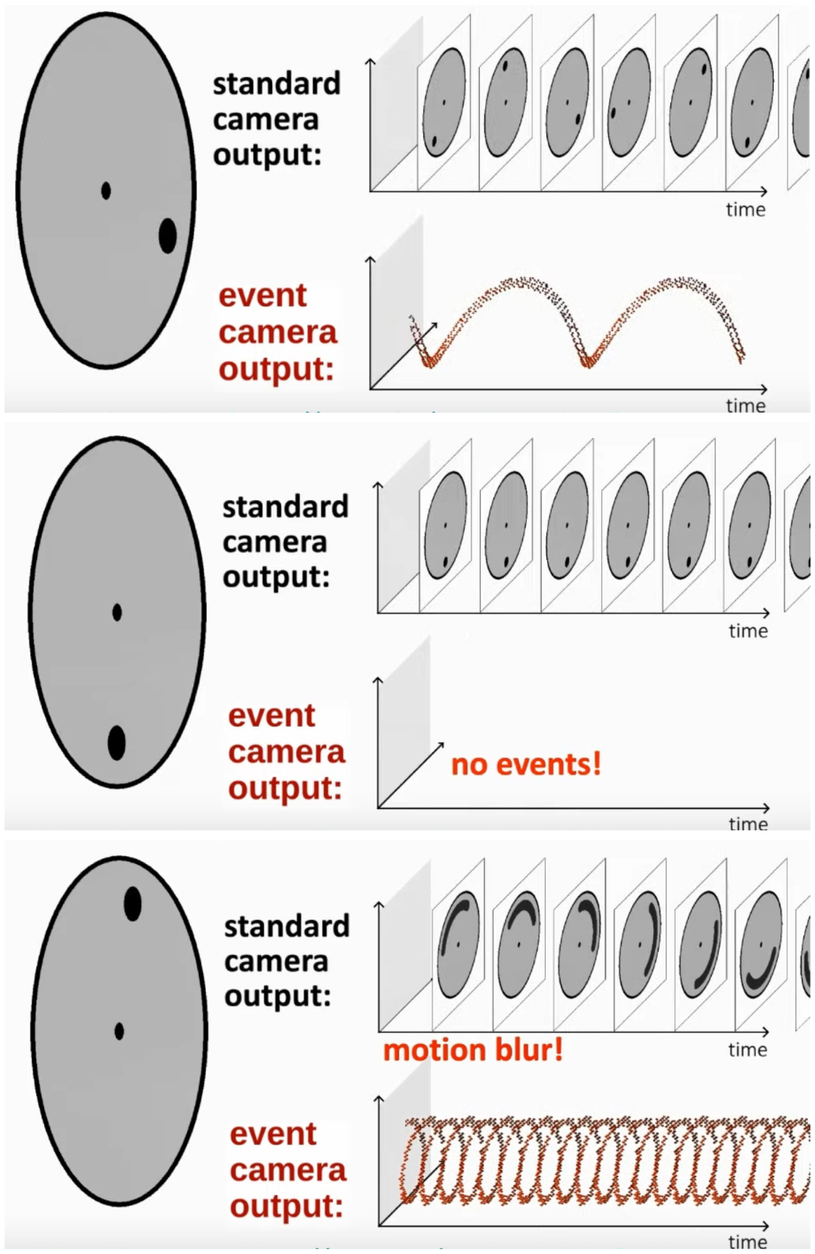


Figure 5. Event camera functionality: Whenever there is motion in the scene, an event camera outputs a stream of low-latency events that map logarithmic intensity changes in a scene (Top). Whenever there is no motion, no events are generated (Middle). Whenever there is a highly dynamic scene, standard cameras tend to produce motion blur at their output. Thanks to their high temporal resolution, event cameras do not show this negative effect (Bottom). Figure adapted from [54].

However, traditional vision algorithms cannot be directly used on an event stream because of their asynchronous and binary nature. Therefore, a paradigm shift is to process an event-camera output in order to take advantage of the properties of this novel sensor.

Given their recent success in computer vision, one of the most important challenges is to find an effective method to apply neural networks originally designed for conventional cameras to the output from event cameras. In other words, how do we convert an event stream into an acceptable input for a neural network? Several methods have been proposed, the most trivial being simple accumulation of events into a frame-like object for a fixed amount of time, or a fixed number of events [64]. Nonetheless, by applying this method we lose all the temporal information and sparse nature of events. Other methods that preserve some temporal information are low-pass filter [74] and the leaky-surface [11]. Currently, the most efficient way to approach this is to convert the event stream into a discretized volume of event in (x, y, t) or voxel grid [104, 24]. Each voxel contains the sum of the polarities that fall into it. Other methods that aim to provide a neural network-compatible input representation include area-count [47] and Time-Ordered-Recent-Events [4].

There is active research in traditional computer vision tasks adapted to work directly from an event stream. Some of these tasks include feature detection and tracking [23], optical flow estimation [104], SLAM [83], visual inertial odometry [66, 64], recognition [11, 61] among others.

Another approach is to reconstruct intensity information from the event stream and then rely on traditional computer vision algorithms to perform the required task. E2VID [67] proposes a recurrent neural network that reconstructs intensity frames directly from the data. The events are first transformed into a fixed-time or fixed-number voxel grid [104] before going into the network. Although this is considered state of the art in intensity image reconstruction, its performance is not real-time. FireNet [75] builds on this by proposing a light-weight GRU-based neural network that achieves similar reconstruction results with faster inference time.

An important reason for why computer vision has had enormous advances in recent years is due to the abundance of publicly available datasets. Due to the novelty of dynamic vision sensors, there is vast deficiency of event-based data. As a consequence, several event simulation tools have been developed to help with the increase the demand of event data. In [17], Hu et al. propose *v2e*, a toolbox that that produces synthetic DVS event streams from intensity frames. This simulator is tested in object recognition and object detection tasks. In this work, we leverage the use of *v2e* to generate an event camera dataset that will serve as input data for training and testing a neural network.

Further avenues of research include the use of bio-inspired spiking neural networks (SNNs) to further take advantage of the sparse and asynchronous nature of events. However, research in this area is still in early stages and results are not yet comparable with those of standard neural network algorithms.

Chapter 3

FPGA IMPLEMENTATION OF ADAPTIVE SUBSAMPLING FOR ROI-BASED OBJECT TRACKING

In this chapter, I present the FPGA implementation of an adaptive subsampling ROI-based tracking algorithm. The aim of this work is to accelerate said computer vision algorithm on an embedded device in order to develop an energy-efficient standalone system. I introduce relevant work, the approach and algorithm description as well as its implementation and software results. I also present the details on the hardware implementation and hardware results. Finally I talk about our attempt of adding an actual ROI-capable image sensor to our system and the difficulties we encountered during the process [1].

3.1 Introduction

There is a wide array of computer vision applications that feature object detection and tracking at their core [52, 12, 13, 16, 3, 60].

Surveillance, autonomous driving, drone navigation are among a myriad of applications that demand low-latency and high-precision tracking. Recent developments in the deep learning domain has inspired researchers to exploit neural networks for these tracking applications [57, 80, 95, 105, 26, 76, 94]. However, algorithm latency is often compromised for the sake of accuracy when it comes to real-world deployment

[1]This material was originally presented in a paper submitted to an IEEE journal and available on Arxiv at https://arxiv.org/abs/2112.09775

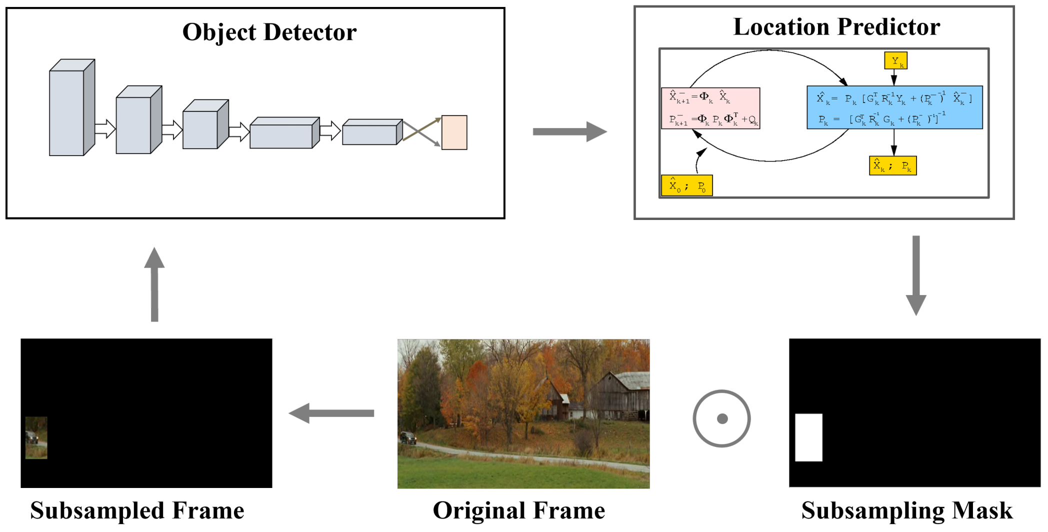


Figure 6. An object detector identifies the ROI and this information is fed as external measurements during the Kalman filter's update phase. The Kalman filter then makes predictions while the non-keyframes are read out from the image sensor.

of neural network driven trackers. For instance, the FCNT tracker [86] achieves a remarkable AUC score of 0.599 post-deployment but is bottlenecked by its latency performance - it only manages to go up to 3 FPS. In a similar vein, the MDNet tracker [55] also achieves an excellent AUC score of 0.678 on the OTB100 dataset, but is let down by its latency performance (it only manages to attain a speed of 1 FPS).

To overcome this problem, researchers are now shifting their focus to energy and resource-efficient tracking solutions. Embedded systems are fast becoming popular platforms for deploying such real-time neural network-based tracking frameworks. These embedded vision systems not only ensure energy efficiency, but also preserve task accuracy. Case in point, the recently proposed SkyNet [99] achieves an impressive IoU score of 0.716 while operating at 25.05 frames per second and 7.26 W power on an Ultra96 embedded FPGA. Embedded systems forsake generality and thus, ensure high task fidelity and efficiency via hardware customization attuned to task needs. As such, embedded computer vision is gaining traction for low-power, real-time vision applications like tracking [92, 96, 58]. However, visual data processing comes with a set of complications posed by system constraints. Typical image sensor readout architectures are notoriously power-hungry [44, 45] and this is a huge bottleneck for developing energy-efficient vision algorithms for real-world applications. To address this issue, there has been increasing research efforts geared towards joint optimization of embedded systems and image sensors for reducing overall system power consumption [100, 27, 59, 29].

Opting for embedded computing environments provides us with the freedom and flexibility to develop and implement highly custom mechanisms targeted towards energy conservation. One such mechanism is based on the notion of region-of-interest (ROI). The ROI is defined by a bounding box enclosing the area of an image frame in which the target object exists. ROI-based energy optimization entails the discarding of pixels outside of the ROI and selectively reading out only the pixels comprising the moving object. The resulting benefits are two-fold: faster bandwidth and improved energy efficiency (owing to selective readout) [73, 87].

Further, the reduction in pixels also has implications in the post-processing stage, where fewer pixels imply fewer clock cycles in the ISP pipeline and in the end vision task. Fewer pixels also alleviates the computational burden and frees up on-board memory and resources to be used up for other tasks. ROI technology can also potentially increase the frame rate of the camera. However, the key challenge now is to adapt existing tracking algorithms for ROI-capable image sensors. We refer to this new class of tracking algorithms as adaptive subsampling algorithms.

In this work, we present an extensive study on adaptive subsampling algorithms featuring various object detectors (both classical and machine learning-based) and we evaluate their performance in an adaptive subsampling setup. We also evaluate the deployment of a Kalman filter as an ROI predictor to help improve the subsampling performance of these detectors by correctly predicting the future object trajectories and making decisions accordingly.

Further, we aim to accelerate these algorithms on FPGAs and study their performance. FPGAs are widely used for low-power and high-latency applications and therefore, are the perfect candidates for implementing energy-efficient adaptive subsampling. Comparing FPGAs to GPUs, we must consider the high power consumption of the latter and the reconfigurability of the former. The obvious choice then becomes FPGA acceleration for our particular use case. In order to implement our algorithms, we leverage Xilinx's Vitis AI tools for mapping our neural network-based subsampling algorithms onto the FPGA. We opt for high level synthesis (HLS) over RTL mapping because of the former's ease of use. Our experimental results show the HLS mapping in no way diminishes the expected performance of our algorithms, that is, we manage to achieve real-time performance without having to resort to RTL mapping.

In this work, we show how we can couple off-the-shelf object detectors with a Kalman filter to jointly perform predictive object tracking and adaptive subsampling. This chapter builds on initial work presented in [31], and we extend that work by introducing several new types of object detectors to the adaptive subsampling pipeline, and evaluating these algorithms on more comprehensive test datasets. In addition, we also identify a suitable candidate for hardware acceleration, and map the neural network-based approach (Efficient Convolution Operators for Tracking (ECO) plus Kalman filter) onto an FPGA. We also show hardware acceleration results with the YOLO plus Kalman filter-based method for comparative evaluation of the ECO-based approach. Averaged across our two benchmarking datasets, the YOLO-based algorithm achieves an AUC score of 0.2721 while the ECO-based approach results in an AUC score of 0.4020. This, coupled with the fact that the we get higher power savings with the ECO method, makes the ECO+KF framework the ideal backbone of our adaptive subsampling algorithm. As per our power consumption model, the YOLO-driven approach requires approximately 6 W while the ECO method requires only 4 W. Additionally, the ECO+KF method operates at 19.23 FPS, which is nearly real-time. Although the YOLO-based method is faster, it is also less accurate. In terms of accuracy-latency tradeoff, the ECO-driven approach is the clear winner.

3.2 Approach and Algorithm Description

3.2.1 Video Subsampling and ROI Prediction

Configuring the image sensor on-the-fly such that the pixels outside the ROI are not read out is referred to as adaptive subsampling [31]. Adaptive subsampling when applied in the context of sequential frame capture is referred to as video subsampling. In video subsampling, ROI is read out for one image frame $I(x, y, t)$, and this information is utilized to determine where the object might move in the following frame $I(x, y, t+1)$ at the next time step. To preserve energy, sensor pixels are switched off by employing an algorithmically determined ROI mask $\hat{M}(x, y, t)$. The mask $\hat{M}(x, y, t)$ is of the same shape as the image frame, and it is all '1's inside of the region of interest and all '0's outside of the ROI. The scene at time step $t+1$ undergoes a Hadamard product operation with the ROI mask determined at time step t and the resulting image is referred to as a subsampled image. That is, subsampled image $I_{subsampled}(x, y, t+1) = \hat{M}(x, y, t) \odot I(x, y, t+1)$. Thus, ROI localization is

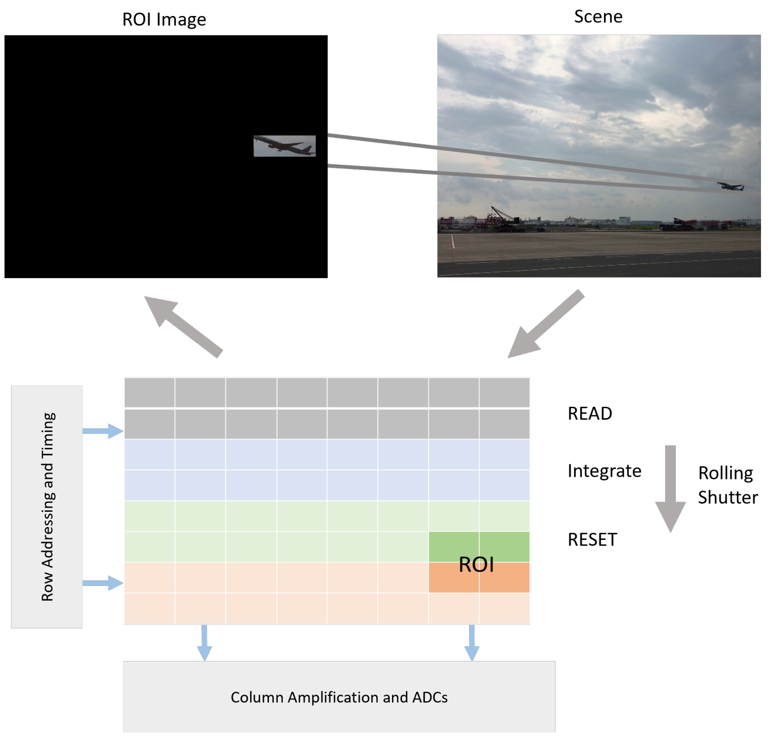


Figure 7. How image sensor reads out an ROI image using a programmable rolling shutter. The rolling shutter mechanism helps capture and integrate temporal and spatial data by adapting the readout timing and exposure length of each pixel row. Here, an ROI image has been obtained using the windowing technique - where pixels outside of the object of interest have been switched off. Energy efficiency can also be improved by skipping (skip every other pixel row or column and thereby read out fewer pixels) and binning (group adjacent pixels together and represent them by a single value, thus resulting in fewer pixels to be read out).

achieved via predictive tracking. It is possible to create ROIs with various geometric shapes like circles/ellipses or even arbitrary shapes. In this work, all our ROIs are rectangular since that is most commonly supported by existing image sensor hardware.

In our video subsampling pipeline, we use a Kalman filter (KF) [34] for making these predictions. It is a recursive estimator that needs only the estimated state from the previous time step and the current measurement to compute the estimate for the current state. It consists of two distinct phases: “Predict” and “Update”.

In the update phase, the difference between the current prediction and the current observation information is multiplied by the optimal Kalman gain and combined with the previous state estimate to refine the state estimate. When this phase requires external measurements to update its state estimate vector, we invoke an object detector $D(\cdot)$ which operates on a fully sampled image frame to localize the ROI. The detector output $D(I(x, y, t)) = b_t$ is a vector containing the bounding box coordinates of the current frame, and this vector b is used as the external measurements in the update phase of the Kalman filter.

The predict phase uses the state estimate from the previous timestep to produce an estimate of the state at the current timestep. Here the Kalman filter solely relies on its own state space matrix and perceptual capabilities to identify the location of the target object. [34] and [32] provide a detailed explanation of how the covariance, observation noise and Kalman gain parameters are computed to update the state space.

Granted, extended Kalman filters, particle filters, etc. might prove to be more accurate and better-suited to our goal of ROI prediction. However, we must also take into account the accompanying resource utilization and on-board clock cycles requirement associated with FPGA acceleration. The Kalman filter is an FPGA compatible and lightweight tracker and, therefore, is ideal for addressing problems relating to latency and computational efficiency. This is why we have chosen the Kalman filter as our prime candidate for ROI prediction.

Notice how the interval between the update and prediction steps of the Kalman filter is programmable. We can delay or prolong the activity of the object detector for as long as we like. We refer to the tracking intervals during which prediction takes place as keyframing intervals. For the Kalman filter-based methods, the keyframing interval would be the interim in which the Kalman filter is made to predict future ROIs without any input from the object detector. If the keyframing interval is defined by k number of frames, then the prediction phase of the Kalman filter will last from time step t till $t + k$. Subsequently, the previously dormant $D(\cdot)$ operator will be activated at time step $t + k + 1$, and a fully sampled image frame $I(x, y, t + k + 1)$ will be fed through it as follows:

$$D(I(x, y, t + k + 1)) = b_{t+k+1}. \tag{3.1}$$

The frames which are readout in their entirety are referred to as key frames and these are the image frames that are sent to the object detector $D(\cdot)$ for processing. In this instance, $I(x, y, t + k + 1)$ is a keyframe.

At the same time step t, the Kalman filter will use the correctional signal from the detector b_{t+k+1}, and undergo the update phase. Thereafter, the prediction phase of the filter will again kick into gear from time step $t + k + 2$ till $(t + k + 2) + k$ - and in the process it will assist in generating subsampled images $I_{subsampled}(t+k+2)$ through to $I_{subsampled}((t + k + 2) + k)$. Thus, the process will continue till the last frame.

The keyframing interval is a primary way to tradeoff between detection and predictive capability of the Kalman filter. The longer the interval, the higher the optimization of the energy and computational efficiency. However, the worse will be the tracking precision as the Kalman filter provides only a simplistic object trajectory model. Hence, it is a matter of the demands of the end task how the precision and energy requirement are to be prioritized. The keyframing interval can be tuned accordingly.

3.2.2 Object Detectors

In our search for a FPGA-compatible object detector that forms the backbone of our adaptive subsampling algorithm, we have studied various off-the-shelf detectors ranging from classical to deep learning-based.

Mean Shift Tracking (MS). The mean shift algorithm leverages the color histogram of an image to keep track of the ROI as a cluster of color histogram values [21]. Note that the notion of coupling the Kalman filter with a mean shift tracker is not a novel concept as evidenced by [71]. However, our algorithm is different in that we set it up as an adaptive subsampling algorithm as in [31]. Also, we conduct a comparative study with other FPGA compatible algorithms and present better methods that outperform this technique.

YOLO and Tiny-YOLO CNN. The YOLO neural network architecture proposed by Redmon et al. [68] is a real-time object detector and is a good candidate for tracking applications. We evaluate both the pre-trained YOLO and the pre-trained tiny-YOLO convolutional neural networks in the measurement phase of the Kalman filter. These networks can optimize the size of their ROI by changing the width and height of the bounding box frame to frame. Additionally, the tiny YOLO's lightweight architecture and its end-to-end optimization framework make it an attractive object detector for mobile and real-time applications.

Accurate Tracking by Overlap Maximization (ATOM). ATOM utilizes high-level target information for offline learning and then employs a dedicated classification component for online learning of object trajectories [14].

Learning Discriminative Model Prediction for Tracking (DiMP). DiMP specializes in leveraging both foreground and background information for ROI estimation [6].

Both ATOM and DiMP demonstrate remarkable tracking capability. However, in our experimental results they do not sustain their performance for adaptively subsampled images. Some adaptations are done to these methods for operating in an adaptive subsampling setup that can be seen at [32].

Efficient Convolution Operators for Tracking (ECO). ECO-based tracking was proposed as a solution for the endlessly increasing complexity of discriminative correlation filter-based trackers [15]. ECO incorporates a factorized convolution operator for reducing the number of filter parameters, a generative model for better characterizing the input data samples and a model update strategy which updates the model parameters after every few frames. We reformulate the aforementioned model update strategy to introduce our Kalman filter in the pipeline for predictive tracking.

	OTB100	LaSOT
MS+KF	0.2051	0.1928
YOLO [68]+KF	0.2709	0.2733
ECO [15]+KF	0.4568	0.3471
ATOM [14]	0.2859	0.2425
DiMP [6]	0.3398	0.2942
Tiny-YOLO+KF	0.0809	0.1103
ATOM+KF	0.4625	0.4282
DiMP+KF	0.4817	0.4702

Table 1. We report the AUC scores with IoU@[0:0.05:1] and keyframing interval of 11 on the two benchmarking datasets - OTB100 and LaSOT.

3.3 Algorithm Implementation and Results

We conduct a set of extensive experiments to evaluate the performance of our adaptive subsampling algorithms in software. These experiments reveal the best candidates for hardware acceleration efforts.

Datasets. We evaluated our joint adaptive subsampling and tracking algorithms on two benchmarking datasets- the OTB100 [90] and the LaSOT [19]. The OTB100 dataset is comprised of 100 test video sequences of varying difficulty. Example tasks include tracking a coupon across a table (easy), a basketball player in a sea of similarly attired sportspeople (medium), and a musician on stage in low-light conditions (hard). On the other hand, the LaSOT dataset constitutes a training set and a test set and has a grand total of 1400 videos. Since we employ pre-trained networks for all of our methods, we only use the test dataset from LaSOT which comprises of 280 video sequences to evaluate our algorithms. Example videos from this dataset include a car moving in a low-traffic road in daylight (easy for a tracking task), an airplane zooming in towards the camera from some distance (medium), and a small drone being flown in random patterns in a park featuring vehicles and other distractors (hard). The frame rates of all videos in our test datasets are 30 FPS.

Metrics. Tracking performance of our algorithms has been evaluated in terms of mean average precision (mAP) and area under the curve (auc) scores. We compute the mAP by counting the number of frames wherein the algorithm prediction and ground truth bounding box have an intersection over union (IoU) greater than some pre-determined threshold. If we perform a sweep over the threshold and plot the corresponding mAPs, we obtain a performance curve referred to as a success plot.

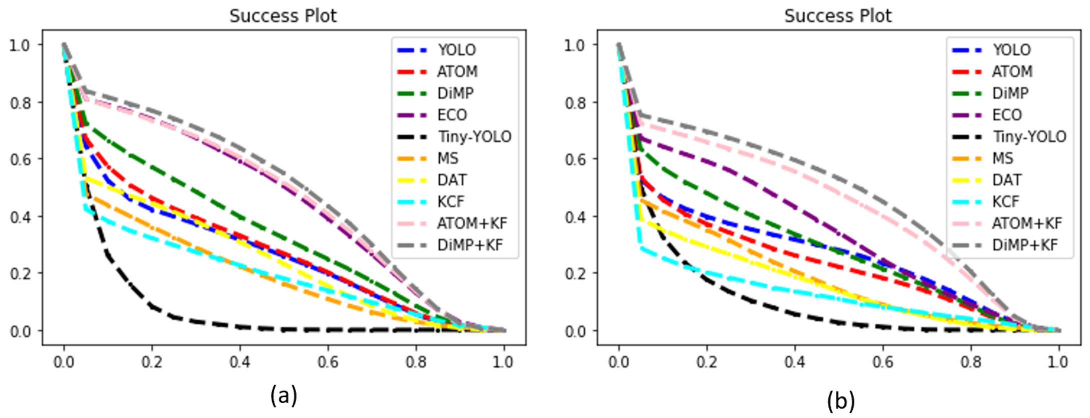
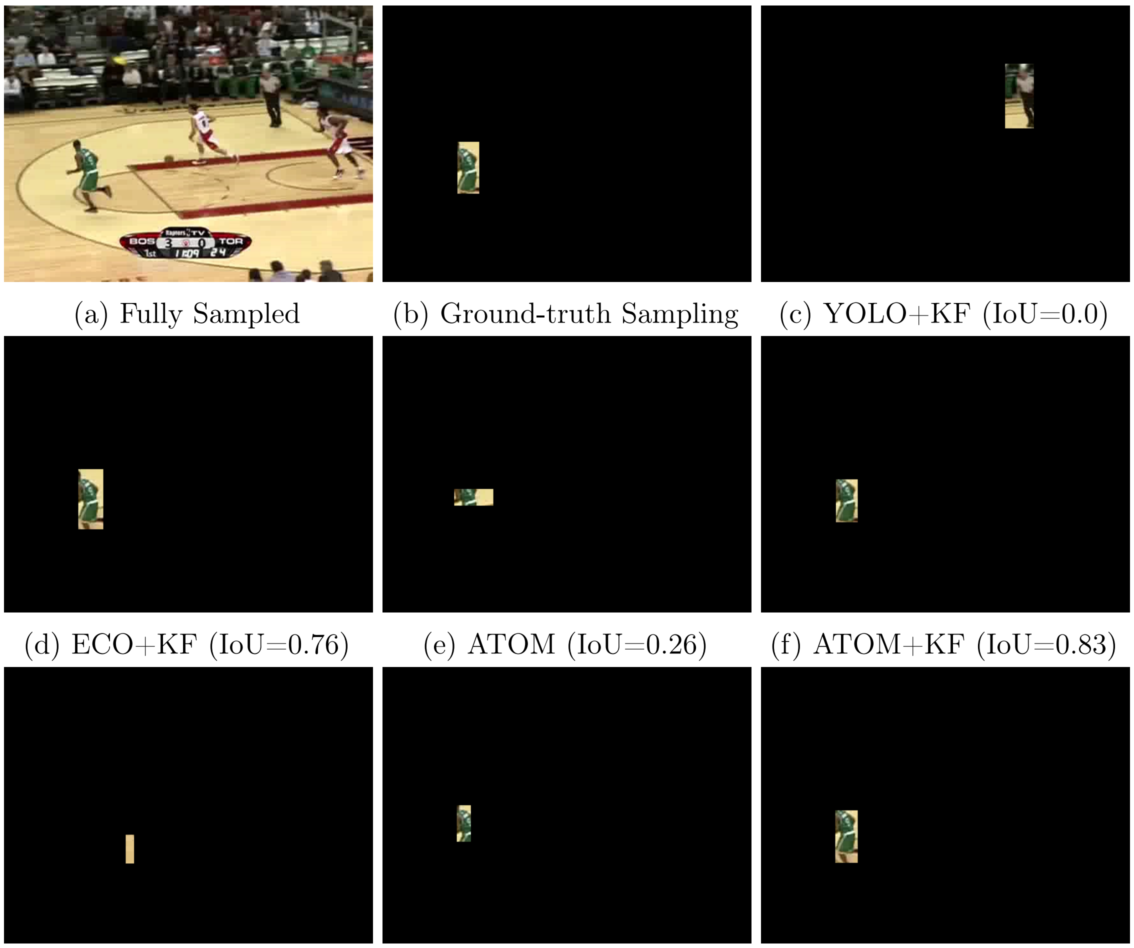


Figure 8. Mean average precision results at different IoU thresholds (success plots). We have swept the IoU thresholds from 0 to 1 with a step size of 0.05 for all the adaptive subsampling algorithms on the (a) OTB100 and (b) LaSOT datasets. It is evident that the methods degrade in performance as the constraint on the IoU threshold is increased.

The area under this curve is the AUC score, and a larger auc score indicates better tracking performance.

Results. In Table 1, we report the AUC scores for the adaptive subsampling algorithms. The DiMP+KF method outperforms the other methods on both the datasets - on OTB100 it attains an AUC score of 0.4817 and on LaSOT it achieves a score of 0.4702. However, the ATOM+KF and ECO+KF can be considered close contenders. As we will later show that the ECO+KF method is best suited for FPGA acceleration and this will be our selected candidate, notice its AUC score on the OTB100 dataset - 0.4568. This is comparable to what the ATOM+KF achieves (0.4625) and is quite close to what we get with DiMP+KF (0.4817). On LaSOT, the ECO+KF demonstrates a little less efficacy (0.3471) but it is still better than other FPGA compatible methods like the YOLO+KF and Tiny-YOLO+KF algorithms. The discrepancy in results for the two datasets may be because of the nature of the captured scenes. Since our focus has been on developing and implementing adaptive


(a) Fully Sampled (b) Ground-truth Sampling (c) YOLO+KF (IoU=0.0)

(d) ECO+KF (IoU=0.76) (e) ATOM (IoU=0.26) (f) ATOM+KF (IoU=0.83)

(g) DiMP (IoU=0.0) (h) DiMP+KF (IoU=0.46) (i) MS+KF (IoU=0.74)

Figure 9. Object tracking and subsampling with the various methods. We select the same frame in a video sequence and display the sensor mask generated subsampled image obtained with our selected approaches. The frame generated by the ECO tracker is visually very close to the ground truth and indicates a good mAP score. Further, the ECO tracker has generated a compact bounding box around the object of interest - indicating that only a few of the pixels in the frame will remain activated during frame readout, thus reducing the power consumption.

subsampling-based tracking algorithms, we simply made use of the pre-trained networks forming the backbone of the object detectors. Training these networks on the training subset of the LaSOT dataset may have mitigated the discrepancies we are seeing from dataset to dataset.

The ATOM and DiMP methods, although considered to be state-of-the-art in the field of tracking, do not perform well in the adaptive subsampling setup without the Kalman filter. Notice how the OTB100 score drops down to 0.2859 from 0.4625 for ATOM when we remove the Kalman filter. In a similar vein, the DiMP score for OTB100 drops down to 0.3398 from 0.4817 - quite a steep degradation. The same is true for the LaSOT dataset as well. This may be attributed to the fact that ATOM and DiMP heavily rely on target-specific information and scene details. When we enforce adaptive subsampling on the input data received by these trackers, there is a distinct dearth of information for the ATOM and DiMP networks to work with and they tend to make erroneous ROI predictions. Compare these to the algorithms where we use the same detectors but in combination with the Kalman filter. In the absence of fully sampled frames at every single time step, the Kalman filter provides tremendous assistance in keeping the trackers within the confines of the correct trajectory. Withdrawing this additional support from the filter makes it very difficult for the ATOM and DiMP methods to sustain their performance. Figure 8 demonstrates the success plots obtained for the various adaptive subsampling algorithms over a range of IoU thresholds. As expected, the greater the constraint on the IoU threshold, the worse the results obtained with the trackers. The tracking performance of these various methods can also be visualized in Figure 9, where the subsampling performance of the algorithms have been compared to the ground truth subsampling mask. The ECO+KF method does an excellent job of honing in on the object of interest while making sure to output a compact bounding box, implying large energy savings for the frame being shown in the figure.

CMOS sensor power analysis. To analyze the power savings that can be achieved with our selected approaches, we conduct a study were we characterize the energy requirements of several CMOS image sensors based on the analysis done in [44]. The algorithm generated bounding boxes are utilized to prompt the image sensor to skip columns not associated with the ROI during frame read out. The idea is to mitigate power consumption by reading out fewer pixels. The power analysis model equations reveal that the average power consumption is proportional to the image resolution [44]. Our study concludes that ECO-based approach manages to attain comparable tracking performance to the best candidates - ATOM+KF and DiMP+KF - while retaining similar power consumption levels as the more energy-efficient trackers. [32] provides details, figures and results regarding this study.

Keyframing, memoization and training on subsampled images. Additional experiments were made to complement the results. An ablation study on keyframing confirmed that a longer interval will have repercussions for tracking performance but will guarantee higher energy savings and faster computation. Memoization is a known strategy in system wherein the results of computationally expensive operations are cached for later use in the event that the same input signals are received in the future. An ablation study on memoization shows that ECO tracker outperforms the other methods on both the OTB100 and LaSOT datasets. It also shows that withdrawing the support of the Kalman filter shows a decline in performance which highlights the necessity of leveraging the Kalman filter to ensure greater tracking fidelity as well as lower latency and higher energy savings. Another study was done to evaluate the performance of these trackers after training them on subsampled images. The study concluded that reducing scene information at training time reduces the learning capabilities of the networks and they fail to predict correct object trajectories at test time. Ref. [32] provides further details on these studies.

3.4 Hardware Implementation

The previous section focused on the analysis and performance evaluation of our adaptive subsampling algorithm in software. The purpose of this section is to discuss the hardware implementation of these algorithms, specifically targeting an embedded device such as a field programmable gate array (FPGA).

Deep learning tasks are usually performed on general purpose computation devices such as microprocessors and graphic processing units (GPUs) which provide a high degree of flexibility but have high energy consumption. On the other hand, application specific integrated circuits (ASICs) are devices dedicated to a single specific purpose that provide ultimate area and energy efficiency at the cost of losing all flexibility [39].

A FPGA is a set of 2D reconfigurable resources that allow mapping of custom hardware architectures. Given their reconfigurable nature, they provide the flexibility of general purpose devices while granting the ability to apply spatial and temporal parallelism, retaining some of the area and energy efficiency of ASICs [20]. FPGAs are also a popular alternative to move IoT applications from centralized cloud-computing environments towards geographically located edge-computing servers [8].

In the past, the implementation of algorithms on FPGAs required the knowledge of Hardware Description Languages (HDL) as Verilog or VHDL. This prerequisite prevented software and algorithm developers from taking advantage of this technology.

However, in recent years several open-source (OpenCL) and proprietary tools (Xilinx Vivado HLS and Vitis, Intel HLS Compiler) have been developed that allow high-level synthesis of applications from high-level programming languages like C++ and Python. In a recent survey [56], an exhaustive list of current and abandoned FPGA HLS tools is presented which is useful reference for the reader.

To accelerate the deep learning component of the object trackers in our adaptive subsampling algorithms we used Xilinx's HLS environment Vitis AI [84]. This is a stack of development tools that allow AI inference on Xilinx FPGAs. It supports mainstream frameworks such as PyTorch and TensorFlow which allow the development of deep learning applications.

Xilinx Ultrascale+ MPSoC edge devices include a processing system (ARM core) and programmable logic (PL) on the same chip. Xilinx provides a synthesized deep learning processing unit (DPU) that maps onto the FPGA, along with a PetaLinux environment that allows the execution of Python scripts and open-source libraries directly on the embedded system. It also supports I/O peripherals like USB and UART. Vitis AI provides Python APIs that allow communication between the programmable logic and processing system. To further accelerate other non-deep-learning components of our algorithm, other tools like the Vitis software platform or Vitis libraries are needed. However, these tools require writing and synthesis of C++ kernels along with the DPU, adding complexity to the system. Given the complexity of this approach, we chose to only accelerate the deep learning section of our algorithm using Vitis AI, while running the rest of our code using standard Python libraries.

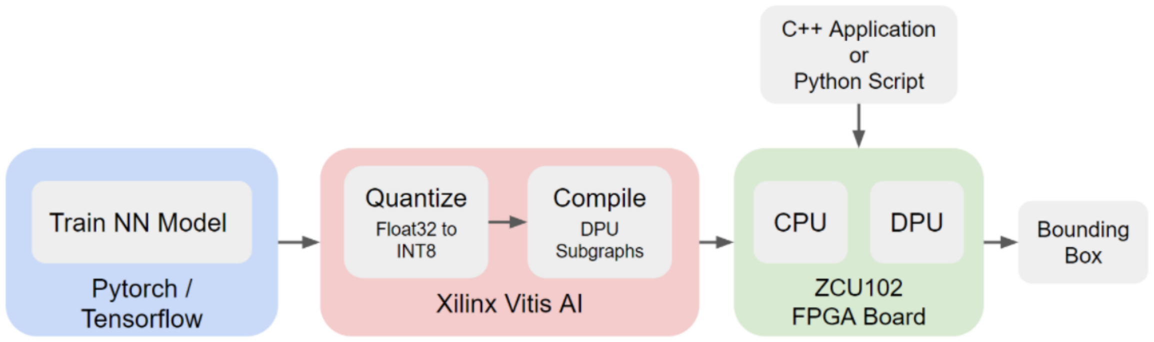


Figure 10. Vitis AI flow.

Figure 10 shows how a deep learning application is deployed on an Edge FPGA using Vitis AI. First, the model is defined and trained offline using one of the supported deep learning frameworks. Next, VAI Quantizer converts weights and activations from 32-bit floating-point to 8-bit fixed point. This reduces computation complexity and memory bandwidth of the model. The quantized model is then passed into the VAI Compiler which maps the network model into a graph-based optimized instruction sequence based on the DPU architecture. The output is a compiled *.xmodel* which will be invoked to run in the programmable logic by VAI Python APIs [91]. One disadvantage of this method is that the input and output size of the compiled models are fixed, therefore we must pre-process our input before sending it for processing to the DPU.

Our hardware setup consists of a Xilinx ZCU102 evaluation board, and a Logitech C920 HD camera connected by USB. The PetaLinux environment is loaded via an SD card, which contains the hardware bitstream, software libraries and application scripts to run the experiments. Figure 12 and Figure 11 display a system diagram and our hardware setup respectively. Within our application, we used OpenCV to capture frames from the camera. Our script also pre-processes the captured frames to fit the model's required input size. We also explored the use of ROI-capable cameras to complete our system. However, we were not able to find a camera that was able to dynamically adjust its ROI while being compatible with our FPGA evaluation board. Given this, we chose to digitally simulate ROI for demonstration purposes as performed in other papers in the literature [38]. The image capture, preprocessing, digital ROI, Kalman filter update and prediction steps, and any other postprocessing required by each specific tracker is done in the processing system of the FPGA board.

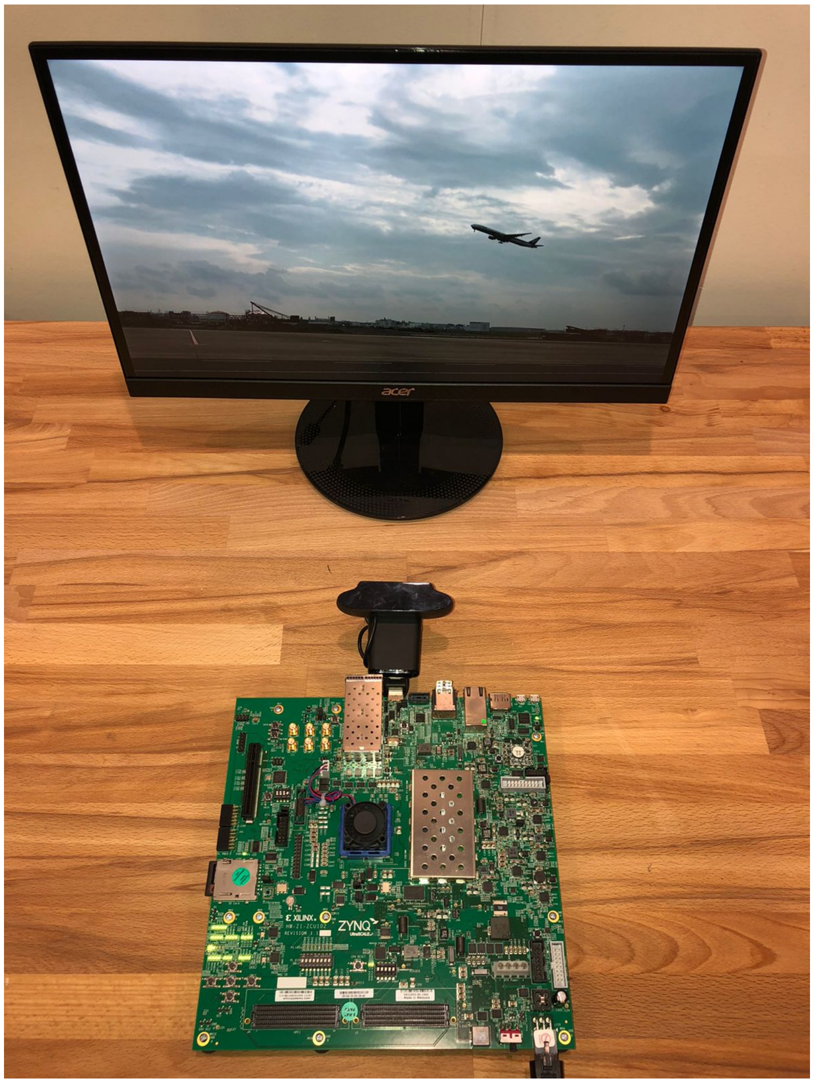


Figure 11. Our hardware setup: A sequence is displayed on an Acer LCD monitor. A Logitech C920 webcam connected to a ZCU102 board captures an image on every keyframe to perform object detection. Our system emulates ROI capture by predicting the bounding box every subsequent frame. This information would be sent to an ROI camera to perform adaptive subsampling.

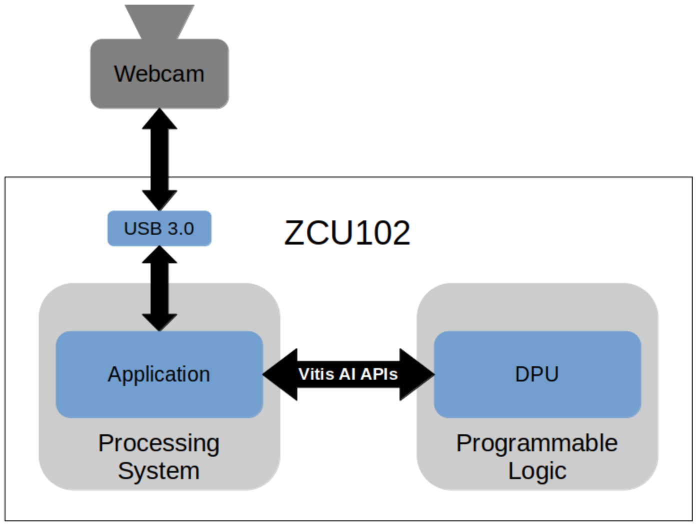


Figure 12. System diagram: The webcam is connected via USB to the board. Our application runs on the processing system while the DPU is mapped on the programmable logic. Vitis AI APIs allow communication between both of them.

As discussed previously, ATOM+KF and DIMP+KF trackers performed slightly better than ECO+KF tracker. However, they made use of custom layers to perform feature extraction. These layers are not compatible with Vitis AI and would require major efforts and HDL knowledge to accelerate and deploy on our system. ECO tracker uses VGG [77] (a classical convolutional neural network architecture) to perform feature extraction. Given its similar performance and layer compatibility we chose to accelerate the ECO + KF algorithm. To implement this algorithm on our system, we accelerated the feature extraction module of ECO following the flow previously described and displayed in Figure 10. This allowed us to run the deep learning component of our algorithm in the programmable logic. The PetaLinux environment of our system allowed us to install and take advantage of popular Python libraries like PyTorch and OpenCV to perform several pre- and post-processing steps on the processing system. It is worth mentioning that some of the libraries used by the original implementation of ECO were not compatible with the ARM core of our board. This forced us to find alternative libraries and perform substitutions to be able to run this tracker. We also chose to accelerate YOLOv3 + KF given that VAI model zoo [85] already provides a pre-trained YOLOv3 model (*yolov3_tf_voc*) and a post-processing library [93]. We incorporated this code into our application and added required camera capture, pre-processing and post-processing steps, as well as the Kalman filter implementation and digital ROI simulation. In our previous work we also accelerated the MS+KF algorithm using SDSoC and xfOpenCV accelerated libraries [31].

3.5 Hardware Results

In Table 2 we show the performance of our algorithm and system. Our algorithm performance takes into consideration the delay of capturing an image, performing pre-processing, detection, post-processing and updating the Kalman filter on every keyframe. While only executing the kalman prediction step on subsequent frames. YOLO + KF achieves a performance of 65.39 frames per second, while ECO reaches a speed of 19.23 frames per second. Although ECO + KF is slower, from the experiments in Table 1 we know that it has a better tracking performance than YOLO + KF. We can therefore conclude that there is a tradeoff between speed and tracking performance of these two algorithms.

Our system performance adds the image capture delay on each subsequent frame. In our current system, this capture delay is identical to the capture delay during keyframes. In a real ROI system, the capture delay of the subsequent frames would be reduced given that we would only need to capture a reduced set of pixels. In this metric, YOLO + KF achieves a performance of 24 frames per second while ECO reaches a speed of 13 frames per second.

In Figure 13 we show a breakdown of the latency at each step for both algorithms. We can see how the image capture, dpu detection and kalman filter steps have a similar latency in both algorithms. However, the preprocessing and postprocessing latency of the ECO tracker is significantly higher than the one from ECO. This is due to the transforms and optimizations that ECO requires to do in its processing steps. These operations are not trivial to implement on an FPGA using the known HLS tools. But since these steps are only done every keyframe, their effect is diminished by using the Kalman Filter to predict the bounding box on subsequent frames.

Algorithm	Algorithm FPS	System FPS	Keyframe
ECO [15]+KF	19.23	13.42	10
YOLO [68]+KF	65.39	24.6	10

Table 2. Performance of our algorithm in hardware.

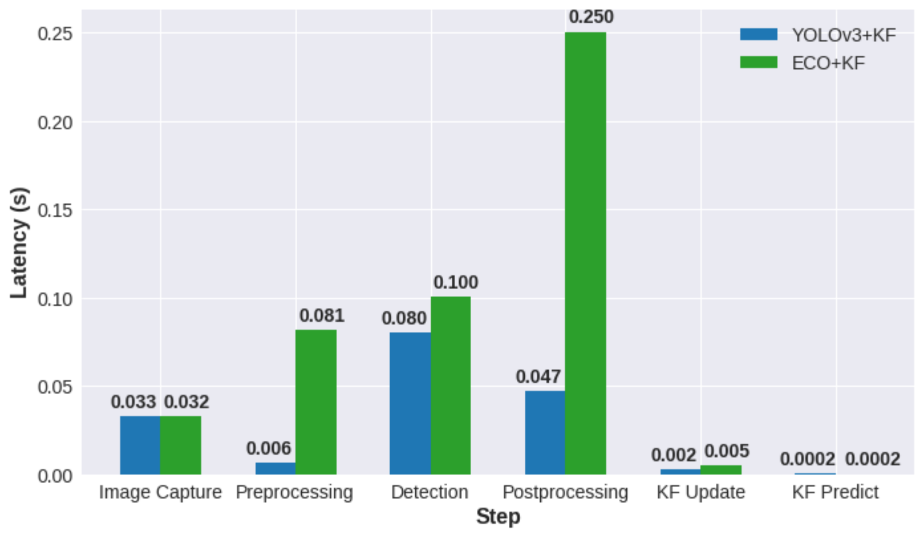


Figure 13. Breakdown of algorithm performance on hardware. Image capture, preprocesing, detection and postprocessing only occur every keyframe. On subsequent frames the Kalman Filter predicts the next bounding box.

The performance of the object detectors can be further increased by multithreading techniques. However, this is only achieved by pre-capturing images so the DPU can work on multiple frames at the same time. For comparison, the YOLOv3 model provided by Xilinx can achieve 34.5 FPS (albeit without taking into account camera capture, pre and post-processing)[85]. To increase our throughput we would need to multithread the camera capture pre- and post-processing, which is not compatible with our algorithm.

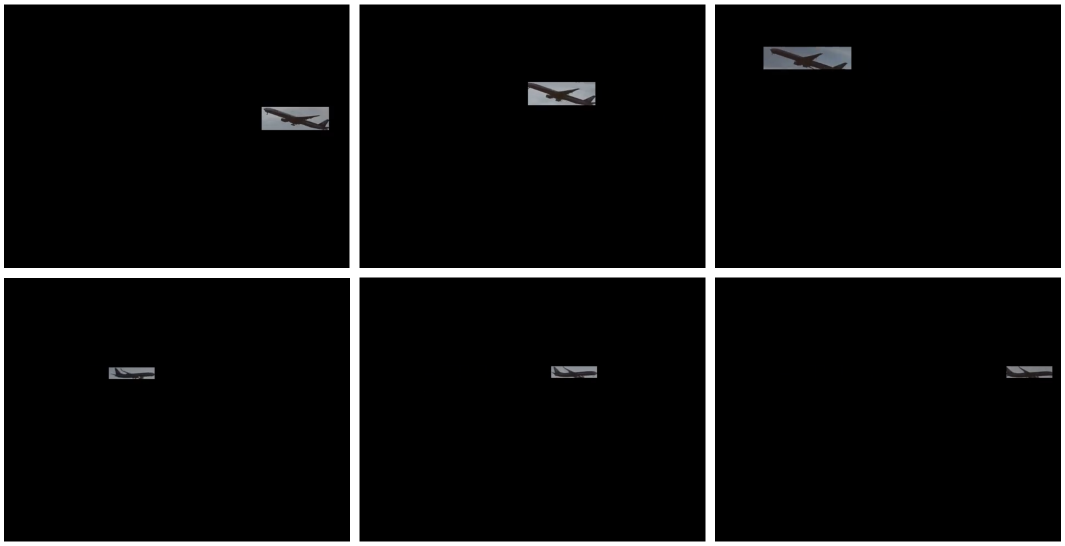


Figure 14. ROI simulation: We simulate adaptive subsampling by masking the area outside the computed ROI. The ROI is obtained by performing object detection every keyframe and Kalman filter prediction every subsequent frame.

As previously mentioned, the DPU is a highly optimized module that performs neural network computations. It is designed to be resource-efficient while performing neural network inference. The DPU has several configuration parameters that can be customized for any specific application. For both of our algorithms, we elected to use the same single-core, 4096-architecture DPU. This configuration uses the fewest possible resources without sacrificing performance on a single-thread. As shown in Figure 15 and Table 3, the DPU uses fewer than 30% of every category of resources. This allows room for other accelerated kernels to be mapped simultaneously on the programmable logic of our system. In our experiments, we found out that increasing the number of DPU cores and therefore the amount of resource utilization, only improved performance when multi-threading the neural network object detection. Since we only do object detection only once every keyframing interval, there were not direct advantages on increasing the number of DPU cores in this algorithm.

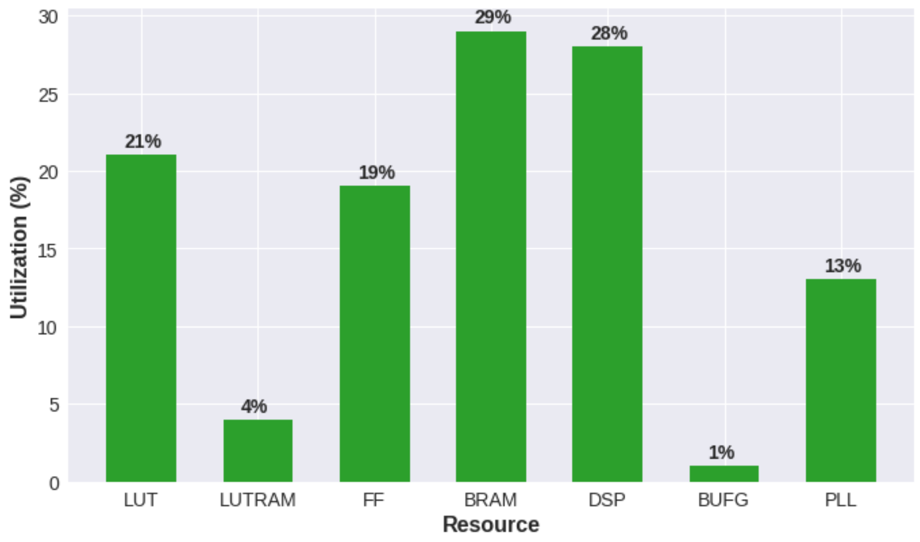


Figure 15. Resource Utilization (%) of a single core DPU architecture.

Resource	Utilization	Available
LUT	58,734	274,080
LUTRAM	6,226	144,000
FF	106,294	548,160
BRAM	261	912
DSP	704	2,520
BUFG	3	404
PLL	1	8

Table 3. Used and available resources on the ZCU9EG FPGA from our ZCU102 evaluation board.

In terms of power, the Vivado post-implementation estimate returns a 9.467 W and 0.779 W of on-chip dynamic and static power respectively for the single-core DPU architecture (including processing system). However, a more fine-grained power analysis is required to compare the savings against a general-purpose CPU or GPU.

3.6 Programmable ROI camera system.

Previously we have shown ROI simulation achieved by masking the area outside the computer ROI as a post-processing step. We also explored performing this masking on an actual image sensor. Several candidates were considered including Avnet's PYTHON-1300 [1], Leopard Imaging's LI-IMX274MIPI-FMC [42] and IDS's UI-3250CP [81].

For PYTHON-1300, we encountered hardware interface difficulties given that the hardware was supported on older Xilinx evaluation boards and Avnet had not developed firmware that allowed connection to our newer ZCU102 board. With Leopard camera, we were able to connect and run provided examples. However, the ROI capabilities could not be exploited due to lack of APIs to access this configurability.

Finally, we decided to use IDS UI-3250CP given that they provide Python APIs and examples for ROI capture. The drawback is that IDS only provides dynamic ROI capabilities in a pre-recorded sequencer mode, meaning that we could only set a limited and pre-configured number of bounding boxes in order to dynamically adjust the sensor readout. The alternative was to re-initialize the camera every predicted or detected bounding box, with the disadvantage of high latency (1.38s) per frame. To run these APIs, we installed Ubuntu 20.04 LTS version on our ZCU102 [25].

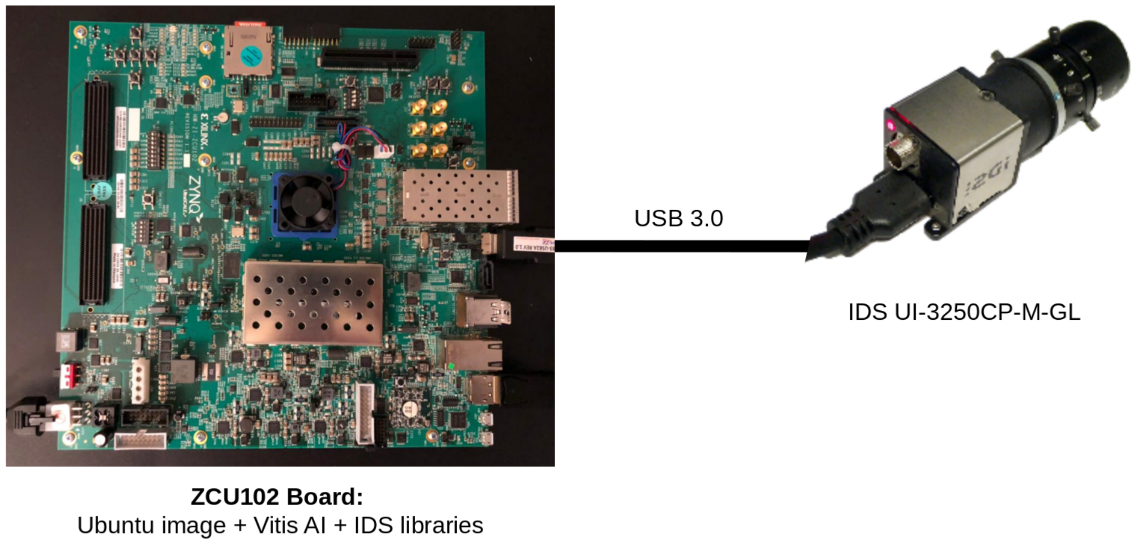


Figure 16. IDS camera hardware setup: IDS image sensor connects via USB3.0 interface. PyuEye [82] provides Python APIs to initialize and control ROI on IDS cameras.

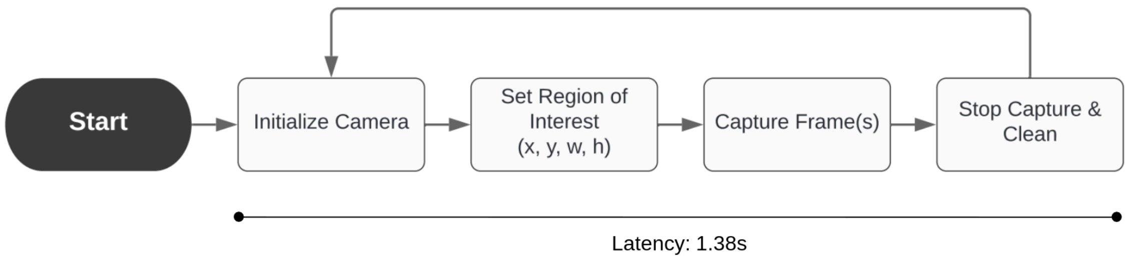


Figure 17. IDS camera functional diagram: Camera initialization is required each time the ROI changes, increasing latency to impractical levels.

3.7 Discussion

This work paves the way for future research in adaptive subsampling-based tracking which has tremendous potential for energy optimization of image sensors. Coupling off-the-shelf object detectors with a Kalman filter results in an efficient mechanism for reconfiguring image sensors on the fly by reasoning about future object trajectories.

We identify an ideal candidate out of these adaptive subsampling algorithms and map it onto an FPGA to demonstrate the feasibility of implementing such methods with a goal to maximizing the energy efficiency.

There is a lot of scope for expanding this work given the current limitations of the proposed approach. Firstly, the predictor in our current framework is a simple Kalman filter. Switching this out with a powerful, state-of-the-art predictor may help improve the tracking precision by leaps and bounds. In addition, the Kalman filter being a classical state estimator requires frequent correctional measurements from an external source (the object detectors in our case). Alternatively, the power of neural networks can be leveraged here to attain superior tracking performance with a network-based predictor. Our intuition is that we may be able to operate at high keyframing intervals with a neural network without compromising the tracking fidelity. Another limitation of our current work has to do with the novelty of the technology. Xilinx has only recently come out with their Vitis AI toolflow and it is still very much in its early stages. We were bottlenecked by the limitations of the Vitis software in implementing the ATOM and DiMP-based approaches. Programming in the ARM core and DPU and transferring data back and forth between the two processing units was also challenging for our particular use case. It required piece-by -piece investigation and restructuring of the modules in the ECO tracker.

Another potential area of improvement for this work is the addition of a performance metric that allows direct comparison with other methods. This metric would have to take into account the system performance, accuracy and energy-efficiency.

Finally, adaptive subsampling and ROI technology are exciting new areas of research with tremendous potential. In this work, we have shown FPGA acceleration as a mode of implementing adaptive subsampling. There is potential to explore ASICs in this area as well in lieu of FPGAs in order to attain better latency and optimized performance. The final step in this avenue of research would be integrate such custom adaptive subsampling algorithms with a real ROI-capable camera sensor, and study the real life performance in terms of latency and precision.

Chapter 4

OPTICALLY DEFOCUSED RECONSTRUCTION FOR EVENT-BASED VISION SENSORS

In this chapter I present a neural network-based pipeline that aims to enhance intensity reconstruction from scenes captured by an event-based image sensor subjected to optical imperfections. The goal of this work is to improve the robustness of these novel sensors, allowing them to be deployed in a wider variety of conditions, including scenes where objects could be present at different focal lengths. First, I introduce the motivation for this work, along with an introduction to the characteristics and applications of event-based cameras. Next, I present the approach and implementation details of our pipeline. Subsequently, I compare results achieved from simulated data against results obtained by capturing data with an actual event-based image sensor. Finally, I present future avenues of work including the potential for lensless event-based sensing.

4.1 Motivation

Event cameras are asynchronous, bio-inspired sensors that measure logarithmic per-pixel brightness changes in a scene. Instead of frames, an event camera outputs a stream of events that encodes time, location and polarity of the logarithmic intensity change at a certain pixel. In contrast with the conventional frame-based cameras, they have superior properties such as high temporal resolution, high dynamic range and ultra-low power consumption [22]. These advantages make them ideal candidates for computer vision applications that demand low-power consumption and latency such as autonomous robotics or always-on sensing.

However, conventional frame-based algorithms cannot be directly used on an event stream. A paradigm shift is needed for the development of novel methods that are able to process their output in order to take advantage of the properties of this novel sensor.

One of the main avenues of research is finding a suitable method of representation that can leverage event-based data as well as decades of previous computer vision research. As previously mentioned in Chapter 2 of this work, several techniques have emerged that try to solve this problem, the most trivial being simple accumulation of events into a frame-like object for a fixed amount of time, or a fixed number of events [64]. Nonetheless, by applying this method we lose all the temporal information and sparse nature of events. Other more advanced methods that preserve some temporal information are low-pass filter [74] and the leaky-surface [11].

Currently, the most efficient way to approach this problem is to convert the event stream into a discretized volume of events in (x, y, t) or voxel grid [104, 24]. Each voxel contains the sum of the polarities that fall into it. Voxel grids have been the intermediate representation of choice for several state-of-the-art event-based algorithms. Tasks such as optical flow [104], intensity reconstruction [67, 75, 10], event simulation [17, 103] and object detection [61]. Other recent methods that aim to provide a neural network-compatible input representation include area-count [47] and Time-Ordered-Recent-Events [4].

Intensity reconstruction pipelines enable us to apply the already existing, mature computer vision techniques to videos reconstructed form a stream of events. Applying this strategy outperforms algorithms that were specifically designed for event data as shown in [65].

Several event integration methods exist in the literature. Some integrate events on the image plane [66, 48, 49] while others integrate them as time surfaces [40, 78, 101, 102]. However, neither method produces natural images, meaning that much of the existing computer vision toolbox, especially deep networks trained on real image data, cannot be applied effectively. Other methods [5, 69, 74] rely on embedding handcrafted smoothness priors to their frameworks. E2VID [67] proposes a network that learns how to reconstruct natural videos from a stream of events. This work achieves reconstructed images that share statistics of natural images through the use of a perceptual loss that operates on mid-level image features [98].

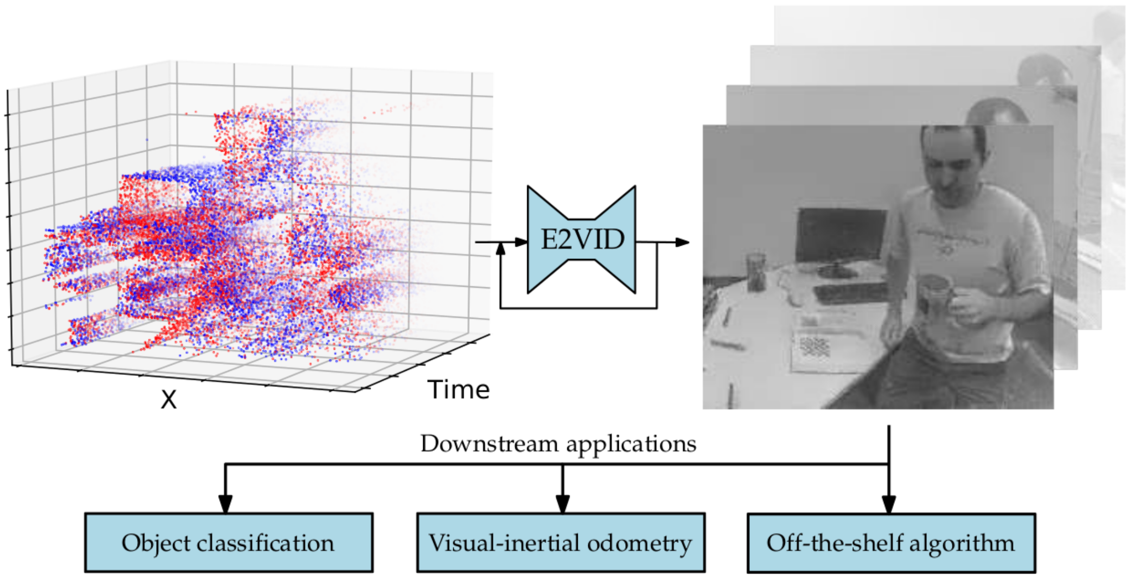


Figure 18. From [65]. E2VID converts a spatio-temporal stream of events into high-quality video. This enables direct application of off-the-shelf computer vision algorithms such as object classification and visual-intertial odometry.

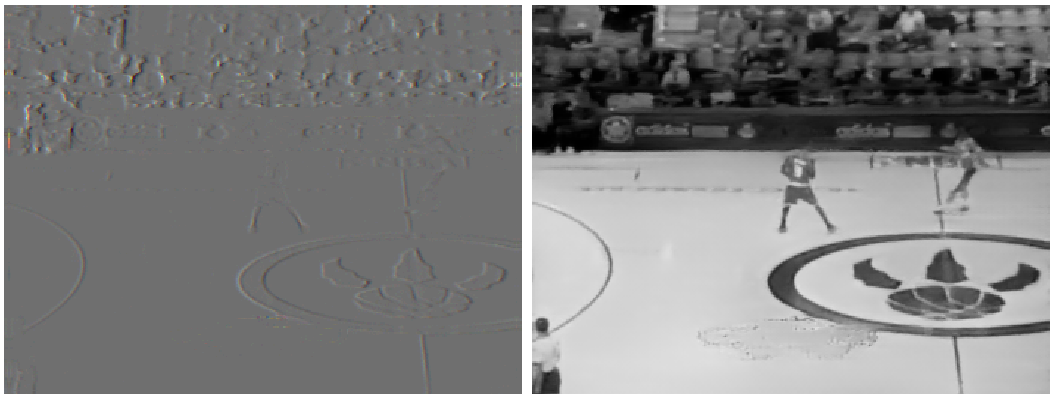
Cameras use lenses to redirect the light rays of a point in the scene to a focused point on the image plane. This carries the advantage of allowing more rays from a single point in the scene being captured by the image sensor. A drawback however, is that only one plane in the scene is perfectly focused onto the image plane.

Current intensity reconstruction models rely on the scene captured with near-perfect focus. Objects outside of the focal plane generate events in an area analogous to the blur circle, instead of a single point which in turn causes the intensity reconstruction model to output sub-optimal natural images. This has a negative effect on the overall performance of the computer vision algorithms relying on them. Figures 19 and 20 show examples on how E2VID fails to properly reconstruct intensity frames from out-of-focus events. Being able to reconstruct intensity frames from defocused events is an important problem that must be solved in order to allow these novel image sensors to be employed in a wider range of conditions.

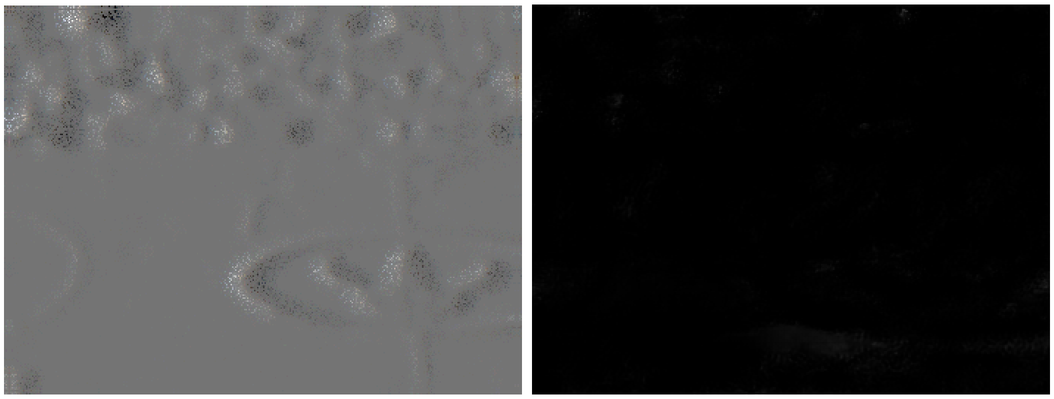
Event-based cameras have high-dynamic range, meaning they can be used in low-light conditions. Increasing aperture size allows more light rays to come into the sensor, with the drawback of reducing the depth of field (DoF). A smaller depth of field causes a wider segment of the scene to be out-of-focus. In order to further take advantage of low-light capabilities of event-based sensors, events should be refocused before being used to perform another task.

To make these sensors more robust, we implement a neural network-based pipeline that learns a mapping to refocus out-of-focus events from an event volume. These refocused event volumes are tested on the existing intensity reconstruction model E2VID. We further improve our pipeline by fine-tuning E2VID.

Groundtruth scene.

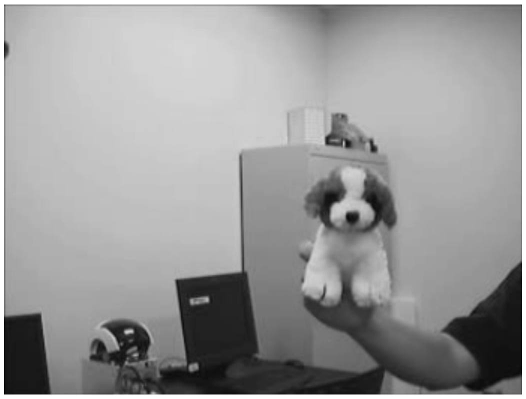


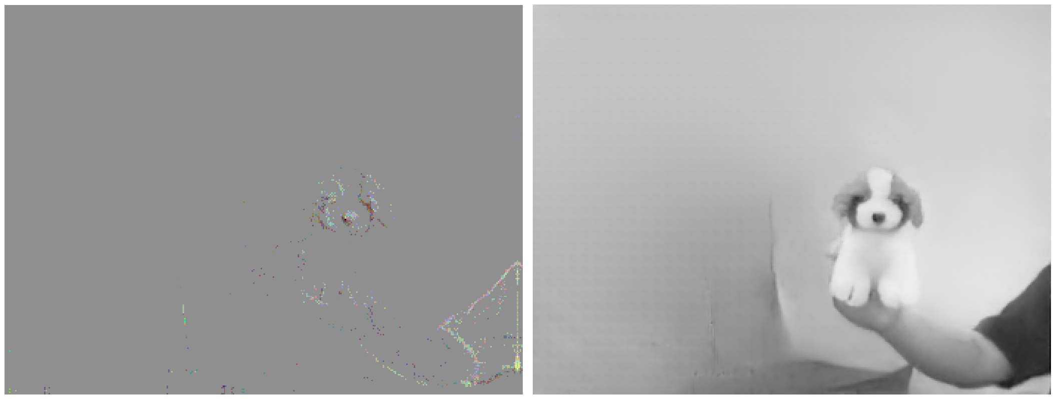
a) Groundtruth voxel (left). Reconstruction from groundtruth voxel (right)



b) Defocused voxel (left). Reconstruction from defocused voxel (right)

Figure 19. Intensity reconstruction comparison: Basketball sequence.


Groundtruth scene.


a) Groundtruth voxel (left). Reconstruction from groundtruth voxel (right)

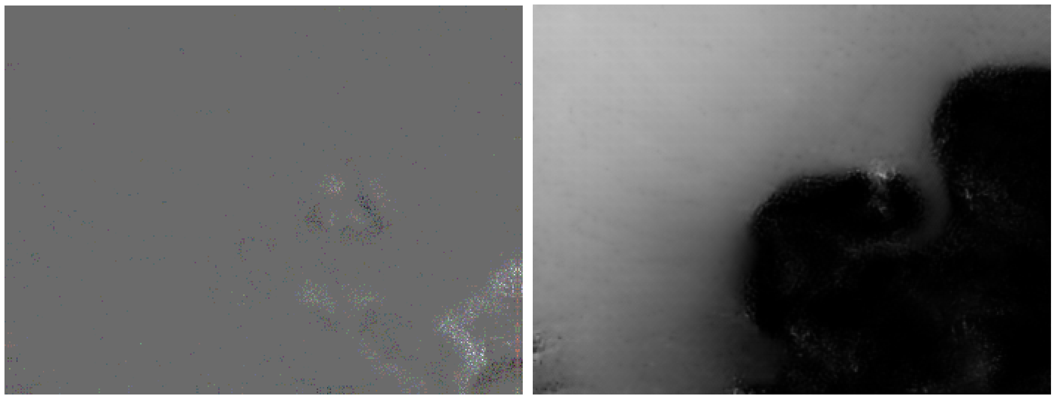

b) Defocused voxel (left). Reconstruction from defocused voxel (right)

Figure 20. Intensity reconstruction comparison: Dog sequence.

4.2 Approach and implementation

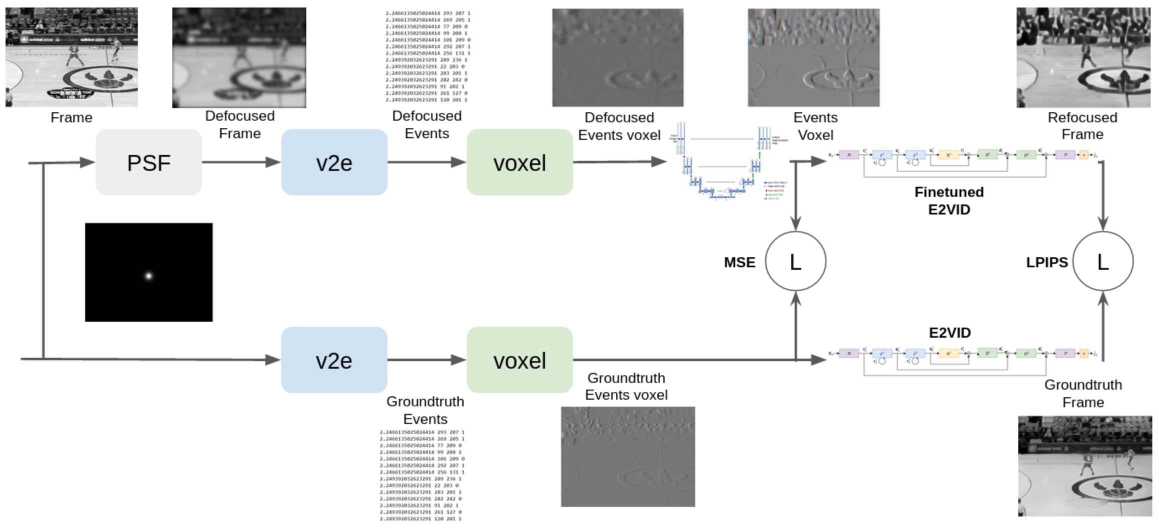


Figure 21. Reconstruction pipeline

Out of focus objects. Out-of-focus events can be approximated as the convolution of a Gaussian point-spread function (PSF) with an object in the scene such that

$$i(x, y) = o(x, y) * psf(x, y) \tag{4.1}$$

where the image i is the result of the convolution of the object o and PSF denoted as functions of position (x, y) in the spatial domain. Since convolution in the spatial domain corresponds to multiplication in the frequency domain, the Fourier transform can be used to convert the space variant function to a frequency variant function as an easier method of implementing the convolution:

$$\begin{gathered}\mathcal{F}\{i(x, y)\} = \mathcal{F}\{o(x, y)\} * \mathcal{F}\{psf(x, y)\} \\ I(u, v) = O(u, v) \times PSF(u, v)\end{gathered} \tag{4.2}$$

where the capitalized notation represents the Fourier transform of the corresponding lower-case function and (u, v) represents the position in the frequency domain. The resulting image in the frequency domain, $I(u, v)$, gives us the frequency and amplitude relationship of the object and PSF, and the spatial-domain image can be recovered by taking its inverse Fourier transform

$$i(x, y) = \mathcal{F}^{-1}\{I(u, v)\} \tag{4.3}$$

Attempting image restoration on degraded images typically means that we're attempting to estimate the true object or scene before any degradation caused by the camera system. This is called the *inverse problem* of the forward model 4.1 and is typically solved using deconvolution methods such Pseudo-Inverse filtering, Wiener filtering or other classical algorithms suchs as *alternating direction method of multipliers* (ADMM). We chose to employ a neural network based approach to solve the PSF deconvolution problem, as done previously in lensless imaging works such as [35].

Event generation. In order to train a neural network however, a large amount of defocused and ground truth event data is required. To generate standard event data as our groundtruth, we passed different video sequences through *v2e* event simulator [17]. We generated simulation datasets employing scenes from object tracking datasets such as LaSOT [19] (1400 sequences) and OTB100 [90] (100 sequences). Before passing the scenes we preprocess them to match the characteristics of an intensity scene captured with the DAVIS camera. This parameters include width (346), height (260) and grayscale conversion. The tool outputs a .txt file that records the events generated by the scene dynamics.

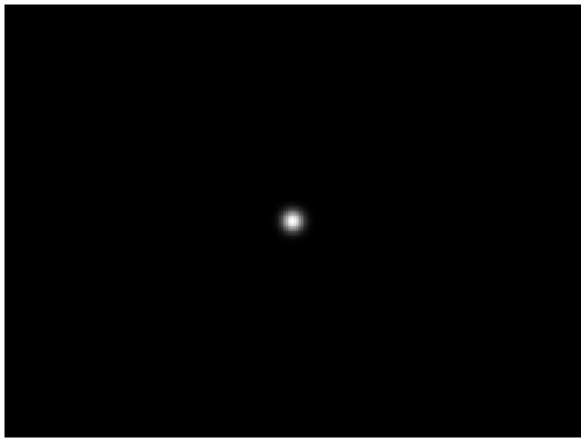


Figure 22. Gaussian PSF used to simulate an out-of-focus scene.

These groundtruth scenes will also be used to generate out-of-focus event data by convolving them with a Gaussian PSF. The convolution is performed by doing a product of the frame by the PSF in the Fourier domain. Finally this scene will be passed to the same event simulator *v2e* to generate a defocused event .txt file.

Event representation. The generated event .txt file contains a list of events that encode the location ($\vec{x}$), time (t) and polarity (p) of each event. To train a convolutional neural network, we need a 3D tensor that contains the information in this data. Different representation alternatives were explored.

Event accumulation consists of accumulating a fixed or variable number of events on a frame-like container. It is possible to accumulate in fixed-event windows or fixed-time windows. For our purposes, fixed-time windows made more sense given that we need to pair groundtruth events with the out-of-focus events. This representation allows training a neural network, but it discards the rich temporal information in the events and is susceptible to motion blur.

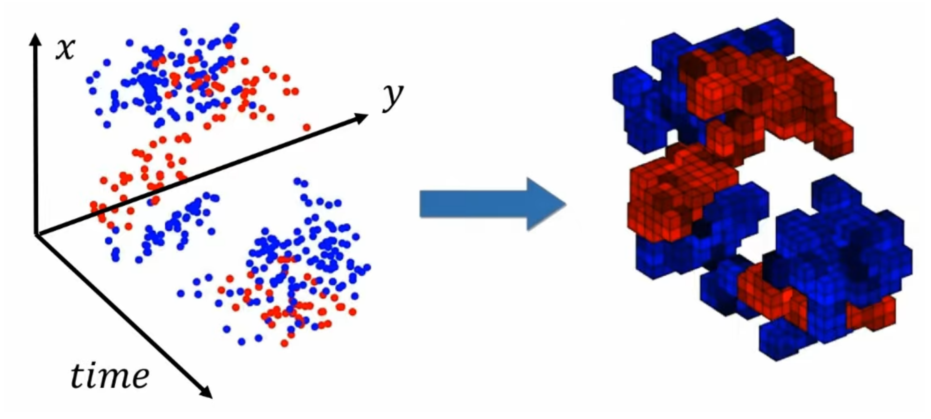


Figure 23. Voxel grid generation: Events are inserted into discretized 3D volumes (x, y, t) or 4D tensor (x, y, t, p) enabling event data to be used as the input to convolutional neural networks [104].

Another alternative is to use a voxel-grid [104] representation. Similar to event accumulation, it takes events during a time-window and inserts them into a fixed number of bins(N). To improve the resolution along the temporal domain, events are inserted using a linearly weighted accumulation similar to bilinear interpolation. This method gives a frame-like 3D representation with the number of bins (N) as the channel dimension that can be used to train a neural network. Another advantage of this representation is that E2VID also uses a voxel grid as the input of its reconstruction neural network. Given this, we chose $N = 5$ to be consistent with E2VID.

Neural Network Architecture. The initial network architecture selected for this project is a U-Net [72] which consists in a decoder and an encoder section along with skip connections between some of its layers, giving it an appearance of a "U". The decoder consists of four downsampling stages. Each stage has two convolutional layers with $3x3$ filters and ReLU activations, batch normalization and maxpooling layer that downsamples the input while increasing the number of channels. The encoder consists in four upsampling stages. On each upsampling stage, the input dimensions are doubled by using bilinear interpolation or a *convtranspose2D* layer, and then does a double convolution to reduce the number of output channels. This architecture has the same shape as the input as in the output. The U-Net used is adapted from open-source Pytorch implementation [51]. We train this network with defocused event voxels as input and event voxels of the same time-window as groundtruth using a Mean Square Error loss and Adam optimizer [37].

Image reconstruction. The output of our network is an N-channels voxel grid. This is the input representation of many different event camera applications like optical flow [104] and image reconstruction [67, 75]. We test our refocusing reconstruction by passing our output and groundtruth through E2VID reconstruction and qualitatively comparing both reconstructions.

After the input voxel is processed through the U-Net based CNN, we added an intensity reconstruction loss that will train the voxel reconstruction network as well as finetuning the intensity reconstruction network (E2VID). The loss used is LPIPS [98], which is measures the perceptual similarity of deep features between two images. This joint training provided the best overall results as will be demonstrated in the next section. Our final pipeline is shown in Figure 21.

4.3 Simulation results

Our model was trained with a set of 1330 simulated sequences from LaSOT dataset [19]. Another set of 70 sequences was selected for testing the generalization ability of the network.

The aim of our refocusing network is to recover the edge information from the out-of-focus event stream. In Figure 24 we can observe how our voxel refocusing network makes the event volume edges sharper.

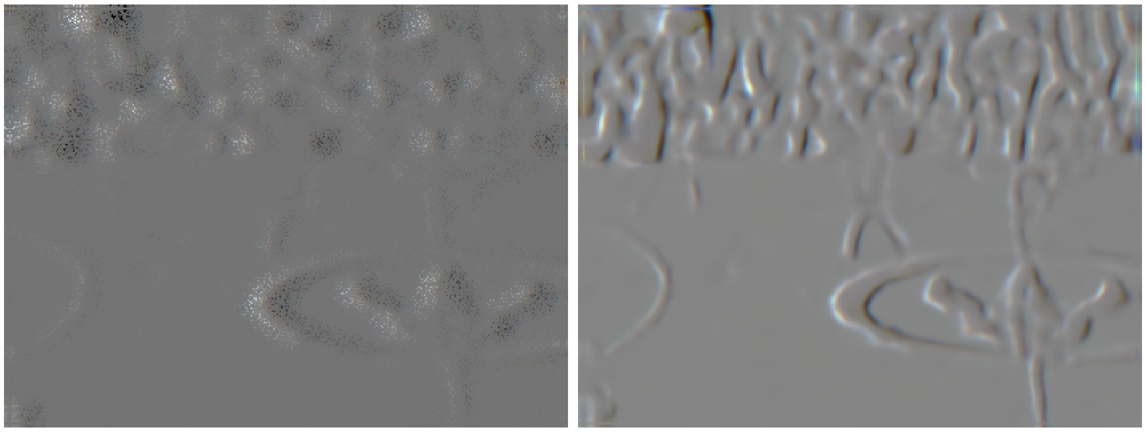


Figure 24. Voxel reconstruction: The first stage of our pipeline refocuses the out-of-focus event voxels (left). The refocused voxels (right) are passed to E2VID for intensity reconstruction.

During our experiments, we discovered that computing and back-propagating the loss from only the refocused event volumes did not result in voxels with optimal statistics for intensity reconstruction. In Figure 25 we can see an example of how the voxels reconstructed by this network do not produce events suitable for intensity reconstruction. Therefore, we decided to add the perceptual similarity loss *LPIPS* [98] at the end of the intensity reconstruction and back-propagate it through both the intensity reconstruction and refocusing networks.

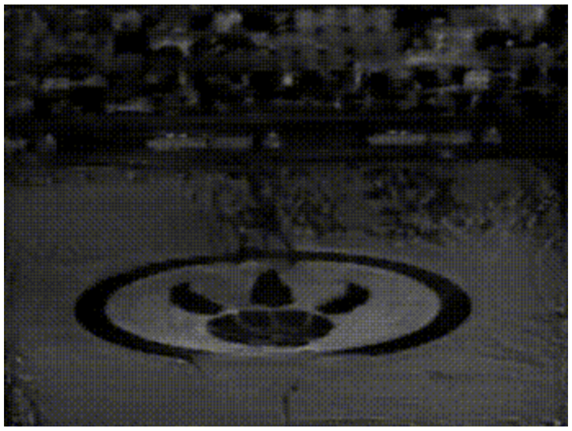


Figure 25. Using MSE as the only loss helped in recovering edge information from the defocused events. However, this did not produced natural image statistics, affecting the intensity reconstruction results.

In Figures 26 and 27 we can see how E2VID is not able to correctly reconstruct intensity information from out-of-focus events (middle). Our event reconstruction network is able to refocus the events and recover edge information (top right). This in turn aids the intensity reconstruction network (E2VID) to recover a clearer intensity frame (bottom right). However, fine-grained and small details are lost after the events are defocused, these details cannot be fully recovered by the event reconstruction network and the intensity reconstruction suffers because of it.

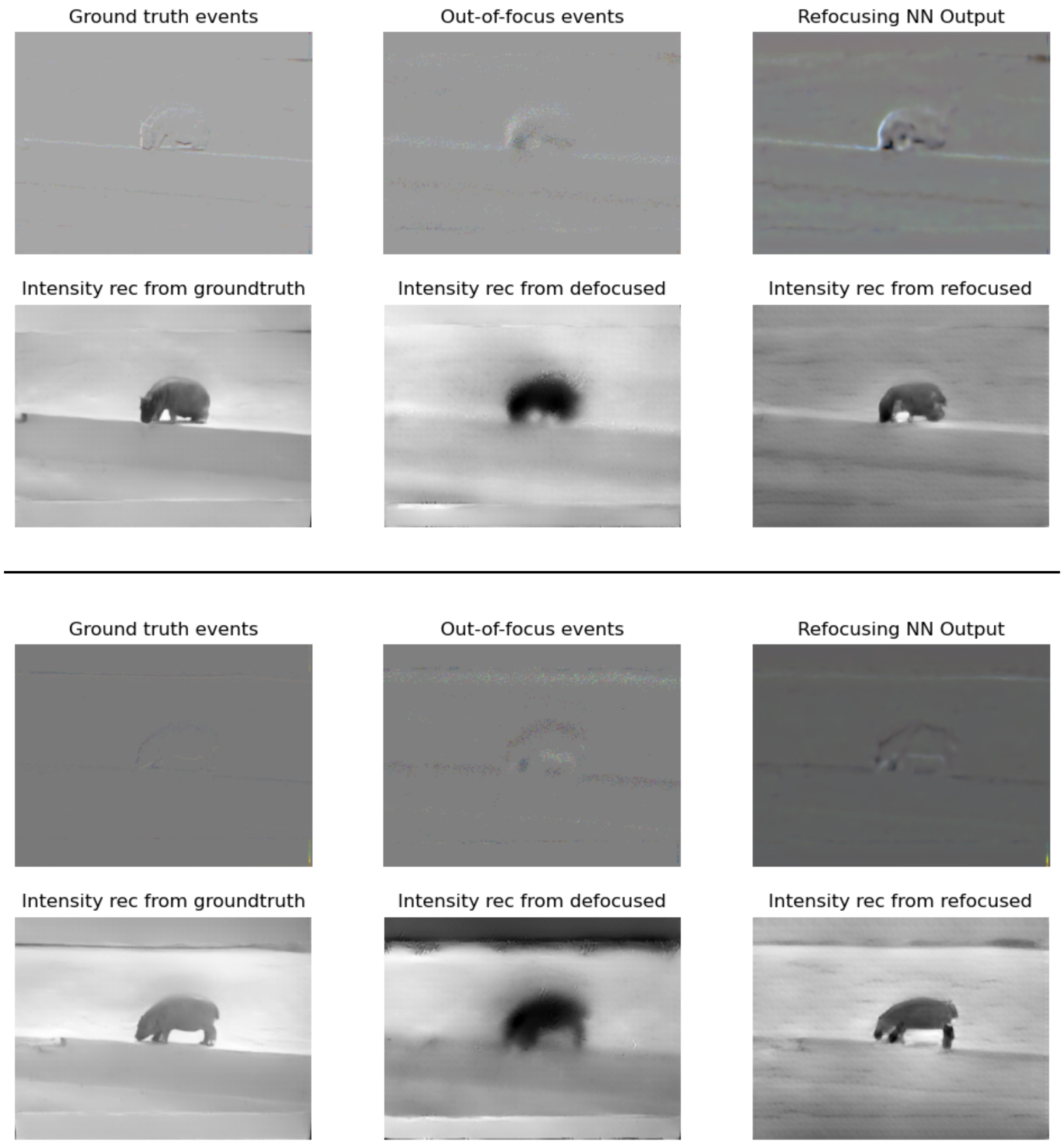

Ground truth events
Out-of-focus events
Refocusing NN Output
Intensity rec from groundtruth
Intensity rec from defocused
Intensity rec from refocused
Ground truth events
Out-of-focus events
Refocusing NN Output
Intensity rec from groundtruth
Intensity rec from defocused
Intensity rec from refocused

Figure 26. Refocusing results: Without first refocusing the events with our network, E2VID cannot correctly reconstruct the intensity to show the scene (middle). When we refocus events with our network (right), E2VID is able to recover most of the information on the scene.

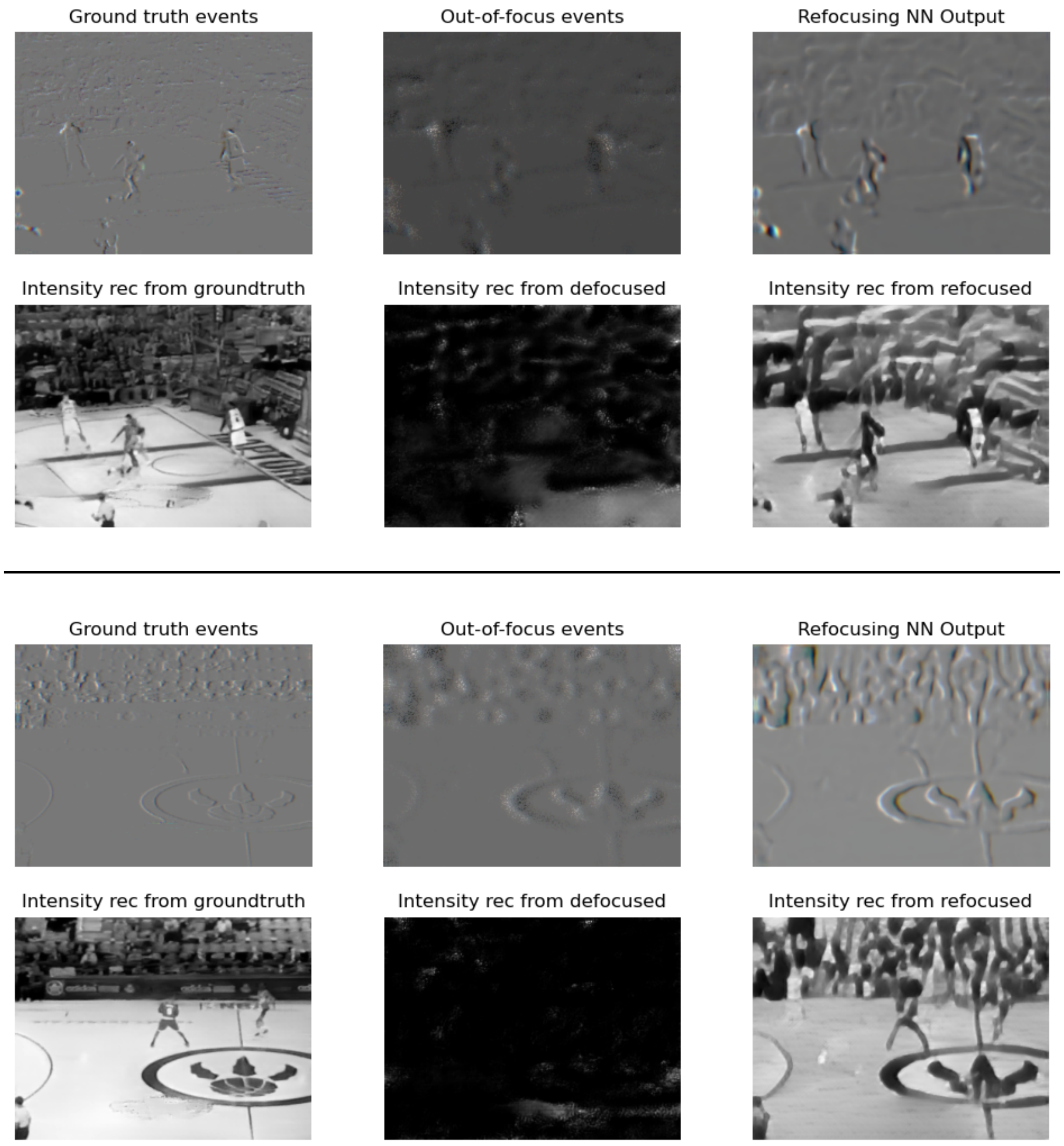

Ground truth events
Out-of-focus events
Refocusing NN Output
Intensity rec from groundtruth
Intensity rec from defocused
Intensity rec from refocused
Ground truth events
Out-of-focus events
Refocusing NN Output
Intensity rec from groundtruth
Intensity rec from defocused
Intensity rec from refocused

Figure 27. Refocusing results: When the scene has high amount of features and movement, E2VID reconstruction on out-of-focus events is worse (middle). Our method allows most of the high-level information to be recovered (right).

The network runs at 61ms latency on an Nvidia GeForce GTX 1060 mobile GPU, meaning that it can achieve up to 16 fps on a mobile GPU without taking into account event capture and representation.

4.4 Real-data results

To test our framework, we captured dynamic scenes with an actual event-based sensor. We utilized the DAVIS 346 event camera from Inivation. This image sensor has the capability to capture both intensity frames and events. We used this advantage to capture different blur point spread functions (PSF) in order to best approximate our simulated PSF. To capture frames we used DV GUI [30], a software for the Inivation Dynamic Vision Sensors that allows capture, visualization and processing of events directly from our camera.

Figure 28. DAVIS 346 Event-based camera: a 346 x 260 resolution image sensor. Has the capability to capture both intensity frames and events.

Figure 29 shows the difference between a real and a simulated PSF. The close up shows that the real PSF does not have exactly a Gaussian shape, meaning the real and simulated do not have the same blur statistics.

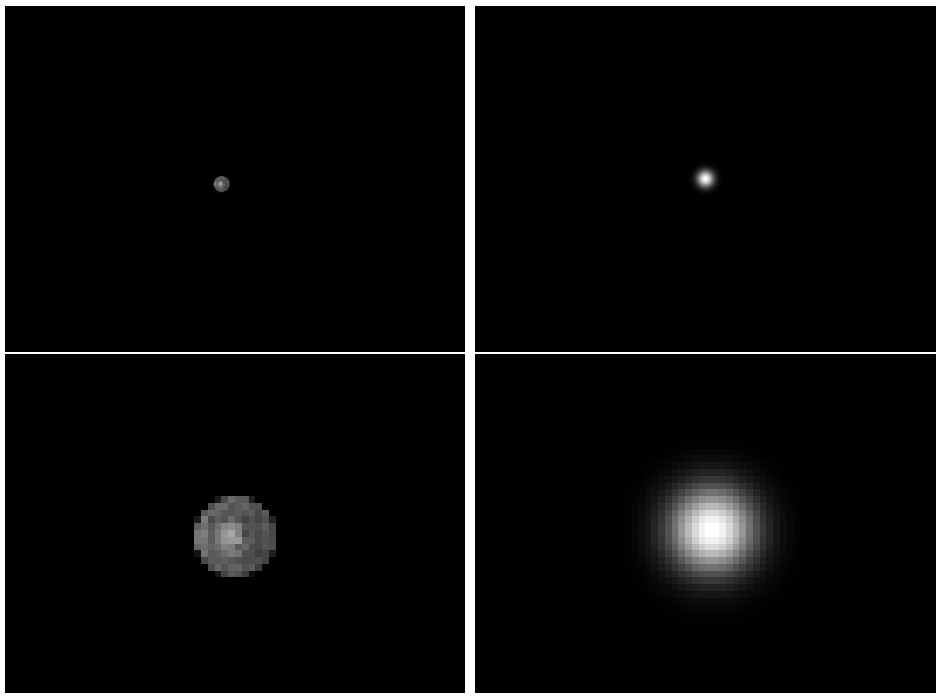


Figure 29. Real PSF (left) vs simulated Gaussian PSF (right) and a close-up to appreciate their differences (bottom).

We generated another dataset from the same sequences (LaSOT, OTB100) but this time using the captured real PSF. We used this dataset to finetune our networks in order to improve the real-data results.

The results on real testing data are shown in figure 30. Here we can observe how our network is in fact refocusing the blurred events to form edges on the event space (top right). This results in better contrast between objects after intensity reconstruction (bottom right).

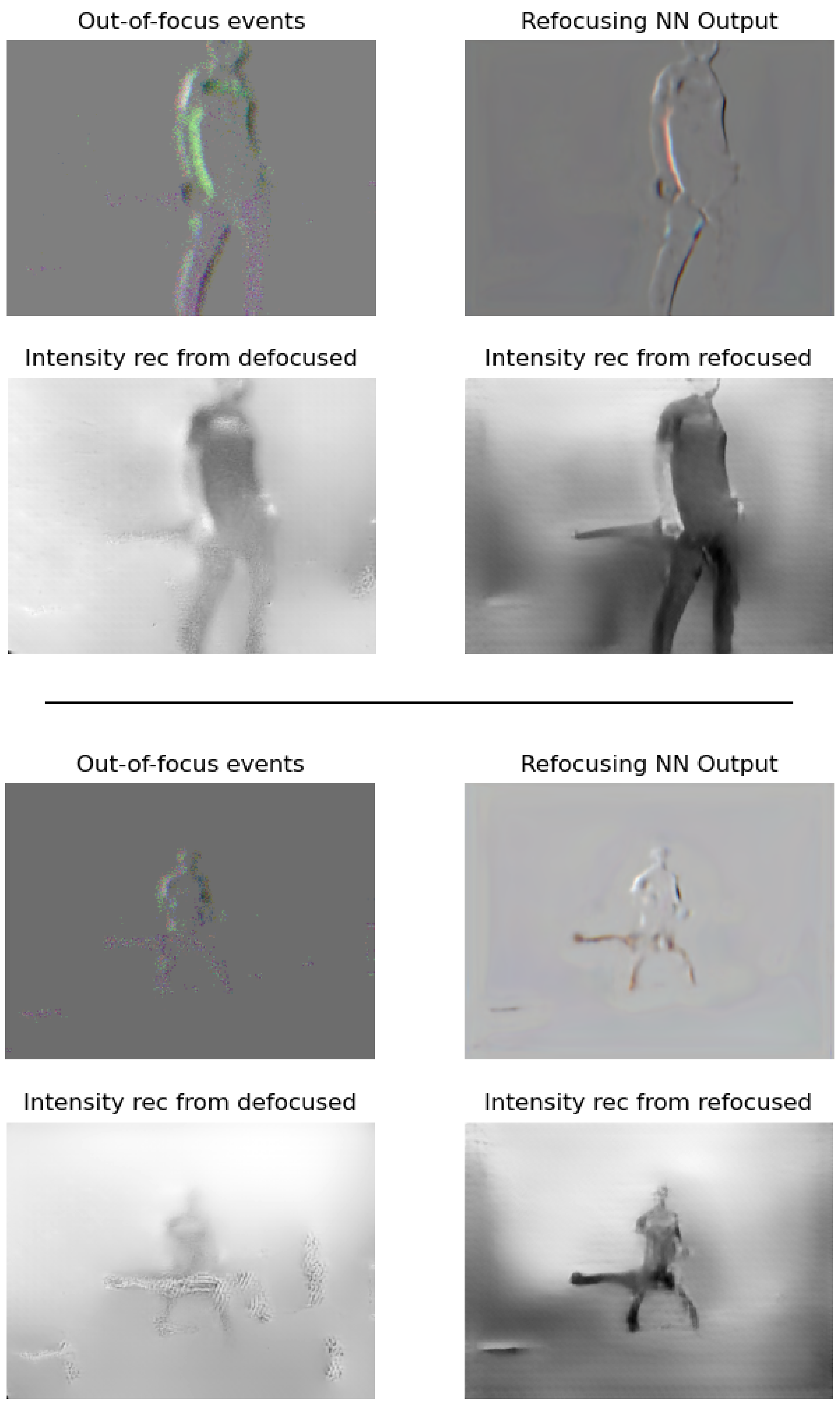

Out-of-focus events
Refocusing NN Output
Intensity rec from defocused
Intensity rec from refocused
Out-of-focus events
Refocusing NN Output
Intensity rec from defocused
Intensity rec from refocused

Figure 30. Results from real data: Our network is in fact refocusing the blurred events to form edges on the event space (top right). This results in better contrast between objects after intensity reconstruction (bottom right).

4.5 Discussion

In this work we have shown a pipeline that makes event-based sensing more robust. By deploying a computational algorithm that deals with optical imperfections, event cameras can be used in non-ideal conditions. With improved robustness, more applications can employ this novel sensors and take advantage of their high dynamic range, low latency and ultra-low power characteristics.

We have also shown how our refocusing pipeline can improve the intensity reconstruction task. However, other tasks such as optical flow and object detection also rely on the voxel grid representation as the input for their methods. In future work, we aim to improve our refocusing network by leveraging different tasks in the literature.

Given that the out-of-focus problem is in essence a PSF deconvolution problem, there is a lot of scope for expanding the current work and taking it to more challenging scenarios. Lensless cameras substitute standard optical elements with thin diffuser materials that multiplex light and form a PSF composed of a caustic pattern [2] instead of a 2D Gaussian. In the future, we aim to expand this work to be able to reconstruct events captured with a lensless event camera.

Chapter 5

CONCLUSION

In this thesis, I presented two different approaches that enable energy-efficient computer vision: offloading computationally heavy algorithms into an FPGA and the usage of low-power event-based image sensors.

In the first approach, we leveraged the use of high-level synthesis tools to accelerate the computationally demanding section of an adaptive subsampling ROI-based tracking algorithm into the programmable logic of a ZCU102 FPGA. This takes advantage of the capability that FPGAs have of enabling temporal and spatial parallelism when mapping custom hardware architectures. We also introduced a working system that implements and executes said algorithm on a standalone embedded device, taking advantage of off-the-shelf hardware, software and libraries.

On the second approach we proposed a pipeline to improve the performance while facing optical imperfections of a novel, low-power image sensor: event-based cameras. By developing an algorithm that can handle imperfections in the optical elements of these sensors, we pave the way for a more robust sensing device that can be deployed in challenging environments. We also generated a dataset to train and test our method by using simulation tools from the literature. In addition, data is captured with an event-based sensor (DAVIS 346) in order to test the performance of our method in real settings.

5.1 Limitations and future directions.

Several opportunities for improvement exist on our FPGA implementation of an adaptive subsampling algorithm. First, only deep learning-based sections of the algorithm were accelerated. In some cases, implementing on hardware non-computationally heavy sections of the algorithm would not bring notable performance increases. Processes such as Kalman filter update and predictions are already very efficient even when executed on the general-purpose processing system. However, ECO+KF implementation uses a Gauss Newton optimization procedure to improve its performance. This algorithm was not accelerated due to the enormous amount of effort required to implement this algorithm in a hardware description language. To further improve the performance of our algorithm, future research could look into accelerating every single section of the algorithm.

Another area of improvement would be that of using a camera that has dynamic ROI capabilities to further improve our system. As mentioned in Chapter 3, we tested several image sensors but none of them allowed to dynamically change the ROI without incurring in long timing delays.

Our event-based refocusing pipeline work intends to be an initial stepping stone in the pursuit of making dynamic vision sensors more robust. While our results on simulated data are promising, once tested on real data our method performance is not comparable to the simulated data. We believe this is because the PSF used to simulate our training data has a different structure than the real PSF. An additional limitation of our method, is that it bases the reconstruction on a single PSF, meaning that objects further out of focus or in focus may not be captured correctly.

Potential avenues of research include improving the refocusing performance of our model by attaching attention mechanisms to our network architecture. Another way to improve our method would be that the network estimates the out-of-focus PSF so the method works independently from the amount of blur in a scene. Finally, a route that was explored without much success but with enormous potential is the prospect of lensless event cameras: removing the standard optical elements and substituting them for a diffusion layer.

REFERENCES

[1] *AES-CAM-ON-P1300C-G.* URL: https://www.avnet.com/shop/emea/products/avnet-engineering-services/aes-cam-on-p1300c-g-3074457345629544173/.

[2] Nick Antipa et al. "DiffuserCam: lensless single-exposure 3D imaging". In: *Optica* 5.1 (Jan. 2018), pp. 1–9. DOI: 10.1364/OPTICA.5.000001. URL: http://www.osapublishing.org/optica/abstract.cfm?URI=optica-5-1-1.

[3] Eduardo Arnold et al. "A survey on 3d object detection methods for autonomous driving applications". In: *IEEE Transactions on Intelligent Transportation Systems* 20.10 (2019), pp. 3782–3795.

[4] R. Wes Baldwin et al. "Time-Ordered Recent Event (TORE) Volumes for Event Cameras". In: *CoRR* abs/2103.06108 (2021). arXiv: 2103.06108. URL: https://arxiv.org/abs/2103.06108.

[5] Patrick Bardow, Andrew J. Davison, and Stefan Leutenegger. "Simultaneous Optical Flow and Intensity Estimation from an Event Camera". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 884–892. DOI: 10.1109/CVPR.2016.102.

[6] Goutam Bhat et al. "Learning discriminative model prediction for tracking". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2019), pp. 6182–6191.

[7] Deepayan Bhowmik and Kofi Appiah. "Embedded vision systems: A review of the literature". In: *International Symposium on Applied Reconfigurable Computing*. Springer. 2018, pp. 204–216.

[8] Saman Biookaghazadeh, Ming Zhao, and Fengbo Ren. "Are FPGAs Suitable for Edge Computing?" In: BOSTON, MA, July 2018.

[9] James Black, Tim Ellis, and Paul Rosin. "Multi view image surveillance and tracking". In: *Workshop on Motion and Video Computing, 2002. Proceedings.* IEEE. 2002, pp. 169–174.

[10] Pablo Rodrigo Gantier Cadena et al. "SPADE-E2VID: Spatially-Adaptive Denormalization for Event-Based Video Reconstruction". In: *IEEE Transactions on Image Processing* 30 (2021), pp. 2488–2500. DOI: 10.1109/TIP.2021.3052070.

[11] Marco Cannici et al. “Event-based Convolutional Networks for Object Detection in Neuromorphic Cameras”. In: *CoRR* abs/1805.07931 (2018). arXiv: 1805.07931. URL: http://arxiv.org/abs/1805.07931.

[12] Zhihao Chen et al. “Real time object detection, tracking, and distance and motion estimation based on deep learning: Application to smart mobility”. In: *2019 Eighth International Conference on Emerging Security Technologies (EST)*. IEEE. 2019, pp. 1–6.

[13] Rita Cucchiara et al. “The sakbot system for moving object detection and tracking”. In: *Video-based surveillance systems*. Springer, 2002, pp. 145–157.

[14] Martin Danelljan et al. “Atom: Accurate tracking by overlap maximization”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 4660–4669.

[15] Martin Danelljan et al. “Eco: Efficient convolution operators for tracking”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 6638–6646.

[16] Yiğithan Dedeoğlu. “Moving object detection, tracking and classification for smart video surveillance”. PhD thesis. bilkent university, 2004.

[17] Tobi Delbrück, Yuhuang Hu, and Zhe He. “V2E: From video frames to realistic DVS event camera streams”. In: *CoRR* abs/2006.07722 (2020). arXiv: 2006.07722. URL: https://arxiv.org/abs/2006.07722.

[18] Heng Fan and Haibin Ling. “Sanet: Structure-aware network for visual tracking”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*. 2017, pp. 42–49.

[19] Heng Fan et al. “Lasot: A high-quality benchmark for large-scale single object tracking”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 5374–5383.

[20] Jeremy Fowers et al. “A Performance and Energy Comparison of FPGAs, GPUs, and Multicores for Sliding-Window Applications”. In: *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*. FPGA ’12. Monterey, California, USA: Association for Computing Machinery, 2012, pp. 47–56. DOI: 10.1145/2145694.2145704. URL: https://doi.org/10.1145/2145694.2145704.

[21] Keinosuke Fukunaga and Larry Hostetler. “The estimation of the gradient of a density function, with applications in pattern recognition”. In: *IEEE Transactions on Information Theory* 21.1 (1975), pp. 32–40.

[22] Guillermo Gallego et al. “Event-based Vision: A Survey”. In: *CoRR* abs/1904.08405 (2019). arXiv: 1904.08405. URL: http://arxiv.org/abs/1904.08405.

[23] Daniel Gehrig et al. “Asynchronous, Photometric Feature Tracking using Events and Frames”. In: *CoRR* abs/1807.09713 (2018). arXiv: 1807.09713. URL: http://arxiv.org/abs/1807.09713.

[24] Daniel Gehrig et al. “End-to-End Learning of Representations for Asynchronous Event-Based Data”. In: *CoRR* abs/1904.08245 (2019). arXiv: 1904.08245. URL: http://arxiv.org/abs/1904.08245.

[25] *Getting Started with Certified Ubuntu 20.04 LTS for Xilinx Devices*. URL: https://xilinx-wiki.atlassian.net/wiki/spaces/A/pages/2037317633/Getting+Started+with+Certified+Ubuntu+20.04+LTS+for+Xilinx+Devices.

[26] Qing Guo et al. “Learning dynamic siamese network for visual object tracking”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 1763–1771.

[27] Sabir Hossain and Deok-jin Lee. “Deep learning-based real-time multiple-object detection and tracking from aerial imagery via a flying robot with GPU-based embedded devices”. In: *Sensors* 19.15 (2019), p. 3371.

[28] Tai-Chiu Hsung and Daniel PK Lun. “New sampling scheme for region-of-interest tomography”. In: *IEEE transactions on signal processing* 48.4 (2000), pp. 1154–1163.

[29] Syed Mudassir Hussain et al. “CMOS image sensor design and image processing algorithm implementation for total hip arthroplasty surgery”. In: *IEEE Transactions on Biomedical Circuits and Systems* 13.6 (2019), pp. 1383–1392.

[30] Inivation. *Inivation DV*. https://inivation.gitlab.io/dv/dv-docs/docs/getting-started.html. Accessed on 2022-04-01.

[31] O. Iqbal et al. “Design and FPGA Implementation of an Adaptive video Subsampling Algorithm for Energy-Efficient Single Object Tracking”. In: *2020 IEEE International Conference on Image Processing (ICIP)* (2020), pp. 3065–3069. DOI: 10.1109/ICIP40778.2020.9191146.

[32] Odrika Iqbal et al. "Adaptive Subsampling for ROI-based Visual Tracking: Algorithms and FPGA Implementation". In: *CoRR* abs/2112.09775 (2021). arXiv: 2112.09775. URL: https://arxiv.org/abs/2112.09775.

[33] Suren Jayasuriya et al. "Software-Defined Imaging: A Survey". In: (2021).

[34] Rudolph Emil Kalman. "A new approach to linear filtering and prediction problems". In: *Journal of Basic Engineering* 82 (1960), pp. 32–45.

[35] Salman Siddique Khan et al. "FlatNet: Towards Photorealistic Scene Reconstruction from Lensless Measurements". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), pp. 1–1. DOI: 10.1109/tpami.2020.3033882. URL: http://dx.doi.org/10.1109/TPAMI.2020.3033882.

[36] Du Yong Kim and Moongu Jeon. "Data fusion of radar and image measurements for multi-object tracking via Kalman filtering". In: *Information Sciences* 278 (2014), pp. 641–652.

[37] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[38] Venkatesh Kodukula et al. "Rhythmic Pixel Regions: Multi-Resolution Visual Sensing System towards High-Precision Visual Computing at Low Power". In: *Proceedings of the 26th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. ASPLOS 2021. Virtual, USA: Association for Computing Machinery, 2021, pp. 573–586. DOI: 10.1145/3445814.3446737. URL: https://doi.org/10.1145/3445814.3446737.

[39] Griffin Lacey, Graham W. Taylor, and Shawki Areibi. "Deep Learning on FPGAs: Past, Present, and Future". In: *ArXiv* abs/1602.04283 (2016).

[40] Xavier Lagorce et al. "HOTS: A Hierarchy of Event-Based Time-Surfaces for Pattern Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39.7 (2017), pp. 1346–1359. DOI: 10.1109/TPAMI.2016.2574707.

[41] Xin Li et al. "A multiple object tracking method using Kalman filter". In: *The 2010 IEEE International Conference on Information and Automation*. IEEE. 2010, pp. 1862–1866.

[42] *LI-IMX274MIPI-FMC*. URL: https://www.leopardimaging.com/product/csi-2-mipi-modules-i-pex/li-imx274mipi-fmc/.

[43] Patrick Lichtsteiner, Christoph Posch, and Tobi Delbruck. "A 128×128 120 dB 15 μs Latency Asynchronous Temporal Contrast Vision Sensor". In: *IEEE Journal of Solid-State Circuits* 43.2 (2008), pp. 566–576. DOI: 10.1109/JSSC.2007.914337.

[44] Robert LiKamWa et al. "Energy characterization and optimization of image sensing toward continuous mobile vision". In: *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*. 2013, pp. 69–82.

[45] Robert LiKamWa et al. "Redeye: analog convnet image sensor architecture for continuous mobile vision". In: *ACM SIGARCH Computer Architecture News* 44.3 (2016), pp. 255–266.

[46] Huibao Lin, Jennie Si, and Glen P Abousleman. "Knowledge-based hierarchical region-of-interest detection". In: *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*. Vol. 4. IEEE. 2002, pp. IV–3628.

[47] Min Liu and Tobi Delbrück. "ABMOF: A Novel Optical Flow Algorithm for Dynamic Vision Sensors". In: *CoRR* abs/1805.03988 (2018). arXiv: 1805.03988. URL: http://arxiv.org/abs/1805.03988.

[48] Min Liu and Tobi Delbrück. "ABMOF: A Novel Optical Flow Algorithm for Dynamic Vision Sensors". In: *CoRR* abs/1805.03988 (2018). arXiv: 1805.03988. URL: http://arxiv.org/abs/1805.03988.

[49] Ana I. Maqueda et al. "Event-based Vision meets Deep Learning on Steering Prediction for Self-driving Cars". In: *CoRR* abs/1804.01310 (2018). arXiv: 1804.01310. URL: http://arxiv.org/abs/1804.01310.

[50] L Marcenaro et al. "Multiple object tracking under heavy occlusions by using kalman filters based on shape matching". In: *Proceedings. International Conference on Image Processing*. Vol. 3. IEEE. 2002, pp. III–III.

[51] Milesial. *Pytorch-UNet*. https://github.com/milesial/Pytorch-UNet. Accessed on 2021-09-22. Aug. 2021.

[52] Bogdan Mocanu, Ruxandra Tapu, and Titus Zaharia. "Deep-see face: A mobile face recognition system dedicated to visually impaired people". In: *IEEE Access* 6 (2018), pp. 51975–51985.

[53] Divya Mohan et al. "Adaptive video subsampling for energy-efficient object detection". In: *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE. 2019, pp. 103–107.

[54] Elias Mueggler, Basil Huber, and Davide Scaramuzza. "Event-based, 6-DOF pose tracking for high-speed maneuvers". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2014, pp. 2761–2768. DOI: 10.1109/IROS.2014.6942940.

[55] Hyeonseob Nam and Bohyung 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.

[56] Razvan Nane et al. "A Survey and Evaluation of FPGA High-Level Synthesis Tools". In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 35.10 (2016), pp. 1591–1604. DOI: 10.1109/TCAD.2015.2513673.

[57] Guanghan Ning et al. "Spatially supervised recurrent convolutional neural networks for visual object tracking". In: *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE. 2017, pp. 1–4.

[58] Paraskevi Nousi et al. "Embedded UAV real-time visual object detection and tracking". In: *2019 IEEE International Conference on Real-time Computing and Robotics (RCAR)*. IEEE. 2019, pp. 708–713.

[59] Daniele Palossi et al. "A 64-mW DNN-based visual navigation engine for autonomous nano-drones". In: *IEEE Internet of Things Journal* 6.5 (2019), pp. 8357–8371.

[60] Youngmin Park, Vincent Lepetit, and Woontack Woo. "Multiple 3d object tracking for augmented reality". In: *2008 7th IEEE/ACM International Symposium on Mixed and Augmented Reality*. IEEE. 2008, pp. 117–120.

[61] Etienne Perot et al. "Learning to Detect Objects with a 1 Megapixel Event Camera". In: *CoRR* abs/2009.13436 (2020). arXiv: 2009.13436. URL: https://arxiv.org/abs/2009.13436.

[62] Murad Qasaimeh et al. "Comparing Energy Efficiency of CPU, GPU and FPGA Implementations for Vision Kernels". In: *2019 IEEE International Conference on Embedded Software and Systems (ICESS)*. 2019, pp. 1–8. DOI: 10.1109/ICESS.2019.8782524.

[63] *Real-Life Use Cases for Edge Computing.* URL: https://innovationatwork.ieee.org/real-life-edge-computing-use-cases/.

[64] Henri Rebecq, Timo Horstschaefer, and Davide Scaramuzza. "Real-time visual-inertial odometry for event cameras using keyframe-based nonlinear optimization". In: (2017).

[65] Henri Rebecq et al. "Events-To-Video: Bringing Modern Computer Vision to Event Cameras". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* June 2019.

[66] Henri Rebecq et al. "EVO: A Geometric Approach to Event-Based 6-DOF Parallel Tracking and Mapping in Real Time". In: *IEEE Robotics and Automation Letters* 2.2 (2017), pp. 593–600. DOI: 10.1109/LRA.2016.2645143.

[67] Henri Rebecq et al. "High Speed and High Dynamic Range Video with an Event Camera". In: *CoRR* abs/1906.07165 (2019). arXiv: 1906.07165. URL: http://arxiv.org/abs/1906.07165.

[68] Joseph Redmon et al. "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2016), pp. 779–788.

[69] Christian Reinbacher, Gottfried Graber, and Thomas Pock. "Real-Time Intensity-Image Reconstruction for Event Cameras Using Manifold Regularisation". In: *CoRR* abs/1607.06283 (2016). arXiv: 1607.06283. URL: http://arxiv.org/abs/1607.06283.

[70] Fengbo Ren. *Energy and area efficiency comparison between different computational devices. From CEN571: Hardware Acceleration and FPGA Computing Spring 2021 slides.*

[71] Jing Ren and Jie Hao. "Mean shift tracking algorithm combined with Kalman Filter". In: *2012 5th International Congress on Image and Signal Processing.* IEEE. 2012, pp. 727–730.

[72] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-Net: Convolutional Networks for Biomedical Image Segmentation". In: *CoRR* abs/1505.04597 (2015). arXiv: 1505.04597. URL: http://arxiv.org/abs/1505.04597.

[73] Priyabrata Saha, Burhan A Mudassar, and Saibal Mukhopadhyay. "Adaptive control of camera modality with deep neural network-based feedback for efficient

object tracking”. In: *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*. IEEE. 2018, pp. 1–6.

[74] Cedric Scheerlinck, Nick Barnes, and Robert E. Mahony. “Continuous-time Intensity Estimation Using Event Cameras”. In: *CoRR* abs/1811.00386 (2018). arXiv: 1811.00386. URL: http://arxiv.org/abs/1811.00386.

[75] Cedric Scheerlinck et al. “Fast Image Reconstruction with an Event Camera”. In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2020, pp. 156–163. DOI: 10.1109/WACV45572.2020.9093366.

[76] Jianbing Shen et al. “Visual object tracking by hierarchical attention siamese network”. In: *IEEE Transactions on Cybernetics* 50.7 (2019), pp. 3068–3080.

[77] Karen Simonyan and Andrew Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *International Conference on Learning Representations*. 2015.

[78] Amos Sironi et al. “HATS: Histograms of Averaged Time Surfaces for Robust Event-based Object Classification”. In: *CoRR* abs/1803.07913 (2018). arXiv: 1803.07913. URL: http://arxiv.org/abs/1803.07913.

[79] Yibing Song et al. “Vital: Visual tracking via adversarial learning”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 8990–8999.

[80] ShiJie Sun et al. “Deep affinity network for multiple object tracking”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 43.1 (2019), pp. 104–119.

[81] *UI-3250CP Rev. 2*. URL: https://www.ids-imaging.us/store_us/ui-3250cp-rev-2.html.

[82] *Using PyuEye*. URL: https://www.ids-imaging.us/programming-examples-details/simple-live-image-acquisition-with-the-python-interface-pyueye.html.

[83] Antoni Rosinol Vidal et al. “Hybrid, Frame and Event based Visual Inertial Odometry for Robust, Autonomous Navigation of Quadrotors”. In: *CoRR* abs/1709.06310 (2017). arXiv: 1709.06310. URL: http://arxiv.org/abs/1709.06310.

[84] *Vitis AI.* URL: https://www.xilinx.com/products/design-tools/vitis/vitis-ai.html.

[85] *Vitis AI Model Zoo.* URL: https://github.com/Xilinx/Vitis-AI/tree/master/models/AI-Model-Zoo.

[86] Lijun Wang et al. "Visual tracking with fully convolutional networks". In: *Proceedings of the IEEE international conference on computer vision.* 2015, pp. 3119–3127.

[87] Yong Wang et al. "Energy-efficient image compressive transmission for wireless camera networks". In: *IEEE Sensors Journal* 16.10 (2016), pp. 3875–3886.

[88] *What is an FPGA?* URL: https://www.xilinx.com/products/silicon-devices/fpga/what-is-an-fpga.html#:~:text=Field%5C%20Programmable%5C%20Gate%5C%20Arrays%5C%20(FPGAs,or%5C%20functionality%5C%20requirements%5C%20after%5C%20manufacturing..

[89] *What is an FPGA?* URL: https://www.intel.com/content/www/us/en/products/details/fpga/resources/overview.html.

[90] Yi Wu, Jongwoo Lim, and Ming-Hsuan Yang. "Online Object Tracking: A Benchmark". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2013).

[91] Xilinx. *Vitis AI User Guide.* English. Version Version 1,4. Xilinx. July 22, 2021.

[92] Wei Yang et al. "An Embedded Tracking System with Neural Network Accelerator". In: *2018 International Joint Conference on Neural Networks (IJCNN).* IEEE. 2018, pp. 1–7.

[93] *YOLOv3 DNNDK.* URL: https://github.com/Xilinx/Vitis-AI/tree/1.3.2/demo/DNNDK/tf_yolov3_voc_py.

[94] Yuechen Yu et al. "Deformable siamese attention networks for visual object tracking". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 2020, pp. 6728–6737.

[95] Mengyao Zhai et al. "Deep learning of appearance models for online object tracking". In: *Proceedings of the European Conference on Computer Vision (ECCV) Workshops.* 2018.

[96] Bingyi Zhang et al. "MiniTracker: a lightweight CNN-based system for visual object tracking on embedded device". In: *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*. IEEE. 2018, pp. 1–5.

[97] Libao Zhang et al. "Multi-scale hybrid saliency analysis for region of interest detection in very high resolution remote sensing images". In: *Image and Vision Computing* 35 (2015), pp. 1–13.

[98] Richard Zhang et al. "The Unreasonable Effectiveness of Deep Features as a Perceptual Metric". In: *CoRR* abs/1801.03924 (2018). arXiv: 1801.03924. URL: http://arxiv.org/abs/1801.03924.

[99] Xiaofan Zhang et al. "SkyNet: a hardware-efficient method for object detection and tracking on embedded systems". In: *Proceedings of Machine Learning and Systems* 2 (2020), pp. 216–229.

[100] Minghao Zhao et al. "Real-time underwater image recognition with FPGA embedded system for convolutional neural network". In: *Sensors* 19.2 (2019), p. 350.

[101] Yi Zhou et al. "Semi-Dense 3D Reconstruction with a Stereo Event Camera". In: *CoRR* abs/1807.07429 (2018). arXiv: 1807.07429. URL: http://arxiv.org/abs/1807.07429.

[102] Alex Zihao Zhu et al. "EV-FlowNet: Self-Supervised Optical Flow Estimation for Event-based Cameras". In: *CoRR* abs/1802.06898 (2018). arXiv: 1802.06898. URL: http://arxiv.org/abs/1802.06898.

[103] Alex Zihao Zhu et al. "EventGAN: Leveraging Large Scale Image Datasets for Event Cameras". In: *CoRR* abs/1912.01584 (2019). arXiv: 1912.01584. URL: http://arxiv.org/abs/1912.01584.

[104] Alex Zihao Zhu et al. "Unsupervised Event-based Learning of Optical Flow, Depth, and Egomotion". In: *CoRR* abs/1812.08156 (2018). arXiv: 1812.08156. URL: http://arxiv.org/abs/1812.08156.

[105] Zheng Zhu et al. "Distractor-aware siamese networks for visual object tracking". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 101–117.

[106] *Zynq UltraScale+ MPSoC*. URL: https://www.xilinx.com/products/silicon-devices/soc/zynq-ultrascale-mpsoc.html.