AI-augmented barcode reader using Xilinx Kria KV260

Alexandru Puşcaşu
Transilvania University of Brasov
Brasov, Romania
alexandru.puscasu@student.unitbv.ro

Ioana Ailenei
Transilvania University of Brasov
Brasov, Romania
ioana.ailenei@student.unitbv.ro

Mihai Miu
Transilvania University of Brasov
Brasov, Romania
mihai.miu@student.unitbv.ro

Coordinator: Cătălin Bogdan Ciobanu
Dept. of Electronics and Computers
Transilvania University of Brasov
Brasov, Romania
catalin.ciobanu@unitbv.ro

Abstract—**This work presents a low-power and cost-effective method for self-service and automated checkout systems in retail, utilizing an Artificial Inteligence (AI)-driven app that uses a camera to scan European Article Number (EAN)-13 barcodes, monitor them, and add them to the bill automatically. The project focuses on employing the Field-Programmable Gate Array (FPGA) capabilities for low-power AI-accelerated systems, targeting the Xilinx Kria KV260. The primary contributions include training and validating a real-time barcode detection neural network, proposing an algorithm for barcode scanning in videos, integrating a convolutional neural network (CNN) on a Xilinx Deep Learning Processor Unit (DPU) for KV260, and creating the first FPGA-based self-service checkout system. The solution addresses limitations of current systems, such as cost-effectiveness and scalability by employing barcodes, cameras, and AI. FPGA technology acts as a dedicated hardware accelerator, reducing energy consumption. The process involves barcode detection, tracking, and scanning in a three-step pipeline, and the experimental setup involves the Xilinx Kria KV260 Vision AI development board, a USB webcam, a display, a USB keyboard and mouse, and necessary software components. The project's impact lies in improving efficiency, cost-effectiveness, and adaptability in retail while utilising AI on reconfigurable hardware. The experimental results suggest that our proposed DPU implementation is 6.1X faster than an NVIDIA Jetson Nano GPU-based implementation.**

I. Introduction

In the last few years, self-check-out and automatic check-out solutions have gained interest in the retail industry. We design a solution that approaches the problem in a new and different manner. Our project consists of developing an AI-based application that uses a camera to read EAN-13 barcodes, track them, and automatically add them to the bill. This project aims to enhance FPGA capabilities for AI-accelerated low-power portable systems using the Xilinx Kria KV260. Using the power of AI, this project aims to streamline processes and improve efficiency in everyday life. Our project is open-source and available on GitHub [1].

The main contributions of this work are:

- Train and validate a neuronal network for detect barcode in real-time;
- Propose and test an algorithm for barcode reading in videos;
- Integrate and develop a CNN on Xilinx DPU, run on KV260;
- Benchmark the proposed approach and compare it with a range of embedded Single Board Computers (SBCs) as well as a laptop and a desktop in terms of performance;
- We implemeneted first FPGA-based self-checkout.

The experimental results suggest that our proposed DPU implementation is 6.1X faster than an NVIDIA Jetson Nano GPU-based implementation.

II. Problem Statement, Motivation

Recently, there has been a great deal of interest in unmanned stores, where we can go in and buy the products we want on the shelf, and when we leave the shop, the payment is done automatically.

There are currently two existing solutions. The first is employed at the Decathlon stores in Europe, which uses Radio-Frequency Identification (RFID) tags on each product. The second solution has been implemented by Intel for a small shop. It uses a network of cameras that tracks the movements of customers and observes which products have been placed in the shopping basket. The disadvantage of existing solutions is that they do not offer a cost-effective, scalable and sustainable solution.

For RFID tags, if we wanted to extend the solution to other retail stores, it would mean sticking a tag on every product, which would lead to price increases. But there is also an environmental and sustainability issue: an RFID tag is a digital circuit with a copper coil to power it. At the moment, there is a global problem with the production of integrated circuits, and we believe companies may decide to manufacture more expensive but perhaps more critical chips such as processors,

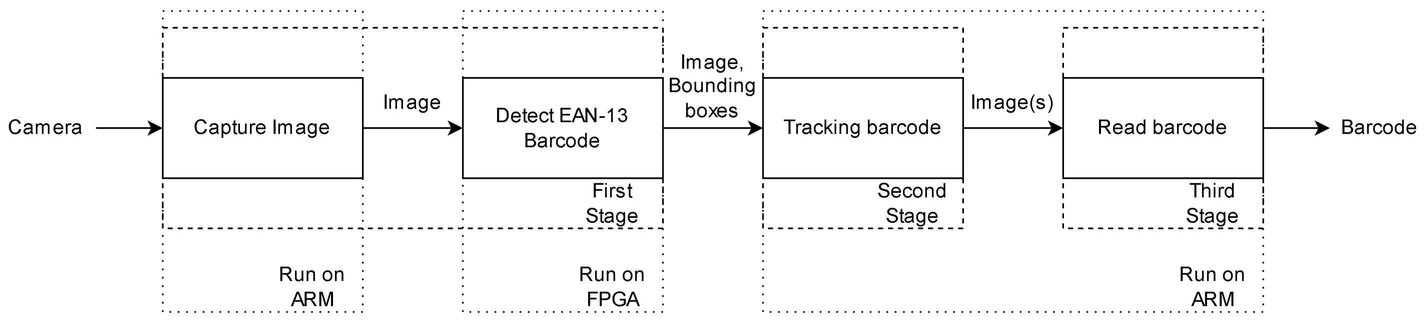

Fig. 1. Pipeline of our approach.

graphics processors, automotive or mobile phone chips. If the companies deploying self-checkout store decide to scale-up this solution, this may also present an environmental and sustainability issue. It will increase the consumption of copper and other materials used in the semiconductor industry, but it will also increase pollution by creating e-waste [2].

The camera array solution has increased computing power requirements, which translates into higher energy consumption. In our opinion, this solution is not scalable for large hypermarket chains due to the large number of cameras required as well as the need for deploying and maintaining the corresponding interconnection infrastructure.

Our proposed solution relies on the barcode printed on every product sold in retail. Each shopping cart will be equipped with two cameras, at diagonally opposite corners, facilitating a 360° view of the shopping cart. The system contains the hardware and software required to be able to detect which products are placed into the shopping cart by scanning their barcodes. Once the barcodes have been read, it is straightforward to access the stores database containing all the product information. Therefore our target system focuses specifically on identifying and decoding the barcodes. The barcode detection must be done in real time and at a reasonable speed. Our solution employs Artificial Intelligence, which in recent years has evolved to offer very good performance and accuracy.

In this project, we employ AI for barcode detection. While traditional solutions used for running real-time AI models are generally not designed to be portable, we will develop an artificial intelligence hardware accelerator. To reduce energy consumption, we propose a dedicated architecture in this project using Field-Programmable Gate Array (FPGAs).

III. Our Approach

We start with an approach based on [3], which proposes to read barcodes with AI in three steps: in the first step, detect the barcode, in the second stage, extract and rotate the barcode, and in the third stage decode the barcode. However, this algorithm has some limitations when using in real-time videos as input. The main problem is that the approach in [3] reads a barcode in every frame, which means that for our specific use case we would have a lot of duplicated values. In order to solve this problem, we have added a tracking step to identify unique barcodes. We tested with different libraries for barcode reading, such as OpenCV, PyZlib, and PyZbar, and discovered that PYZbar does the barcode rotation internally, so we could remove this stage from our pipeline.

In order to have real-time performance on CNN interference, we need dedicated hardware, such as an accelerator. The software solution allow fast development and prototyping. We chose a System on a Chip (SoC)-FPGA, the Xilinx Kria KV260 Vision AI. On the reconfigurable hardware portion of the SoC we have integrated the Xilinx DPU.

The integration was done by employing the Vitis flow. Because we do not find a suitable base platform for KV260, we have designed one, with SoC, interrupt controller and clocks. On the Vitis side we used the platform and within a connection file we have specified where the DPU's clocks and interfaces will be connected. The CNN interference utilises the DPU accelerator instantiated on the reconfigurable hardware while the remaining steps of the proposed solution are executed in software on the ARM cores presend on the SoC.

In Figure 1 we present the pipeline we designed. The pipeline has three stages which are described below: in first stage, we detect barcodes; in second stage, we track them, to find the unique one; finally, in the third stage, we read the barcode. The barcode detection stage runs on the DPU, while the other stages run on ARM cores. The program is wrote in Python and we use PYNQ library to control and communicate with FPGA side.

In the first stage, we trained a YoloV3-Tiny CNN [4] with around 90,000 pictures. To reach real-time performances we use input of size 224×224. We chose this architecture because this CNN has a reduced number of layers that run fast in real-time detection, and the layers are also simple and well supported by the DPU. To populate the training dataset we used ZVZ [5] and Roboflow [6], and to increase the number of pictures in the dataset we have augmented the original pictures by rotating every image 360 times, one degree at a time.

Next, we employ the B4096 DPU accelerator, which is integrated on the reconfigurable hardware. We use a hardware-software co-design approach for this stage: the preprocessing part of the data is done on the CPU, the CNN interference runs on the DPU, and postprocessing is performed on the CPU. We

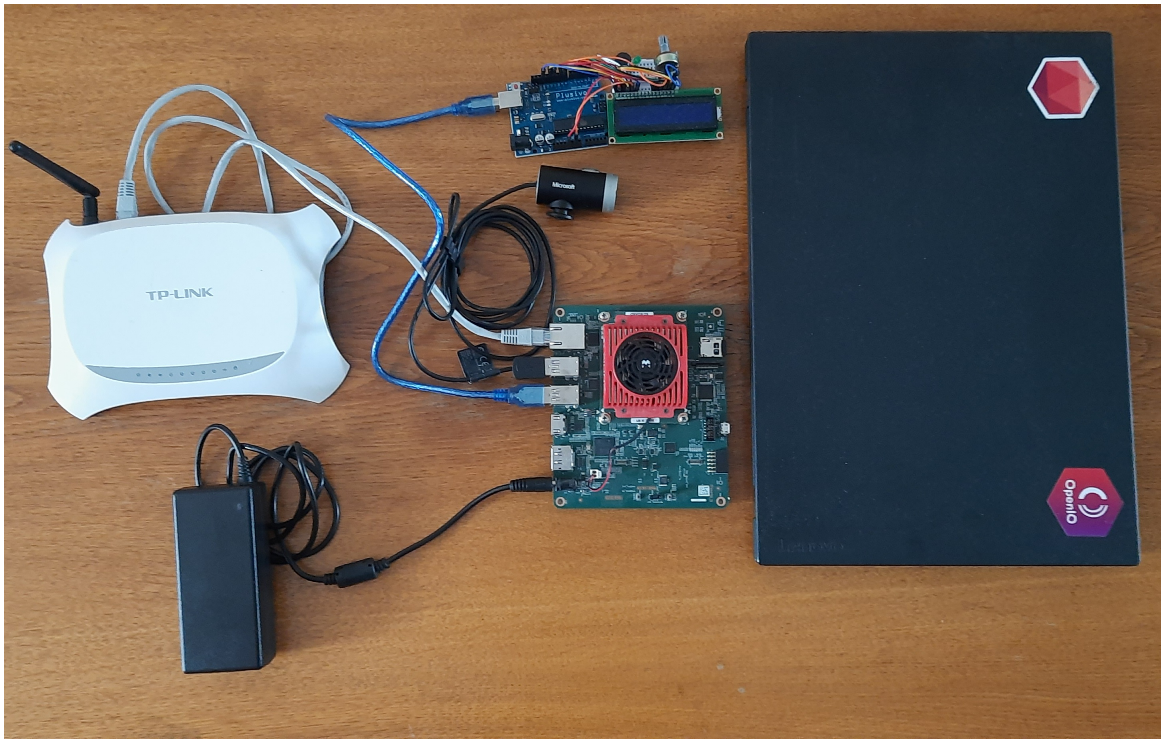

Fig. 2. The system components: Xilinx Kria KV260 Vision AI FPGA board using a 12V, 3A power supply, custom LCD display using an Arduino board, USB webcam, network switch, and a computer

employed the Xilinx Vitis flow to integrate DPU on the FPGA. The hardware architecture is described in Figure 3.

In the second stage, we track the barcodes. For that task, we use the SORT algorithm [7], which has high accuracy with a low computational cost [8]. The algorithm assigns a unique id to every bounding box over the whole app's running time. In this stage, we also have the ability to detect the direction of movement, which allows us to develop features like adding or extracting products from the bill.

In the third stage, we read the barcode exactly one time using the PyZbar library.

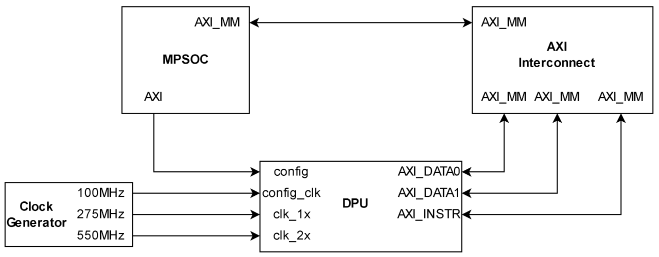

Fig. 3. Hardware architecture - conceptual representation

IV. Experimental setup

In this Section we will describe the experimental setup and present a series of steps for setting up the hardware and software components necessary for the project, including reading data from a USB web camera and displaying the data on other PCs using SSH X11 forwarding as well as a dedicated LCD display integrated using an Arduino board. The hardware used is presented Figure 2.

We use the Xilinx Kria KV260 Vision AI board [9] for this project, which is based on a non-production version of the the K26 System-on-Module (SoM) [10] The board has the following specifications [11], [12]:

- CPU: Quad-Core ARM Cortex-A53@1.33GHz + Dual-Core ARM Cortex-R5F@533MHz
- Cache: 32KB Level 1 instruction cache, 32KB Level 1 data cache, 1MB Level 2 cache
- GPU: ARM Mali-400 MP2
- RAM: 4GB
- 256K LUT
- Distributed RAM 3.5 Mb
- 144 36Kb Block RAM blocks, 5.1Mb total block RAM
- 64 288Kb UltraRAM blocks
- 1.2K DSPs
- FPGA max frequncy: 650MHZ

The hardware components we have used in this project are the following:

- Xilinx Kria KV260 Vision AI development board;
- 12V, 3A, Power supply for the Xilinx Kria KV260 board;
- USB web camera: Microsoft LifeCam Cinema for Business;
- Display monitor or HDMI-compatible screen;
- USB keyboard and mouse;

TABLE I
APPLICATION PROFILING RESULTS

Stage	Core i7-4710HQ[%]	SOC-FPGA[%]	KV260 (ARM)[%]
Detection	91	14.20	94.4
Tracking	0.5	0.54	2.5
Frame wait	6	81.15	1.87
Other	3.5	4.1	1.23

TABLE II
EXPERIMENTAL HARDWARE.

Platform	Processor	Base Freq. [GHz]	Max Freq. [GHz]	Cores	Threads	Arch. [nm]	RAM
KV260	ARM A53	1.33	1.33	4	4	16	4GB DDR4 SDRAM
Raspberry Pi 3B+	ARM A53	1.4	1.4	4	4	28	1GB LPDDR2 SDRAM
Raspberry Pi 4	ARM A72	1	1	4	4	28	1GB DDR4 SDRAM
NVidia Jetson Nano	ARM A57	1.43	1.43	4	4	20	4GB LPDDR4, 1600MT/s
NVidia Jetson Nano	NVidia Maxwell	0.921	0.921	128	4	20	4GB LPDDR4, 1600MT/s
Server	Xeon X3440	2.53	2.93	4	8	45	8GB DDR3 ECC, 1600MT/s
Laptop	Core i7-4710HQ	2.5	3.5	4	8	22	12GB LPDDR3, 1333MT/s

TABLE III
RESOURCES UTILIZATION ON FPGA

Resource	Utilization	Available	Utilization %
LUT	48336	117120	41.27
LUTRAM	5597	57600	9.72
FF	98552	234240	42.07
BRAM	17	144	11.81
URAM	64	64	100.00
DSP	690	1248	55.29
BUFG	3	352	0.85
PLL	1	8	12.50

- Arduino Uno board and necessary components for the display (resistors, wires).

We have used the following software tools in this project:

- Xilinx Vitis 2021.2 software development environment;
- Ubuntu 22.04 (aarch64) for Xilinx KV260 Vision AI;
- Xilinx Deep Learning Processor Unit (DPU) software stack, ver. 1.4;
- OpenCV 4.8 library for image processing;
- Python 3.10 programming language;
- SSH client software for X11 forwarding (e.g., PuTTY, OpenSSH);
- Arduino IDE for programming the Arduino board.

We have followed the following steps when testing our design:

1) Connect the USB web camera to the KV260 development board;
2) Run the developed barcode reader application on the KV260 board;
3) Establish an SSH connection to the remote machine and start the barcode data display application;
4) Verify that the barcode data captured by the web camera is successfully transmitted and displayed on the remote machine or Arduino-based display;
5) Perform testing with different barcode samples and validate the accuracy and performance of the system.

V. EXPERIMENTAL RESULTS

First prototype was done in software and ran on an Intel-base laptop. We decided to made an application profiling, and analyze how much time every stage consume, we record how much every stage use compared with global application run time. We count the detection stage and tracking stage. The profiling results shows that another time consuming operation is frame waiting, we decided to add this operation on our analyze. In Table I we compare profiling results from 3 platforms, one Intel-base computer, the ARM CPU from KV260 and our solution that use SOC-FPGA. The prototype on the KV260 runs 6.4X faster at detecting barcodes than the Intel Core i7 laptop we used. Also, the KV260 platform has to wait for frame reading due to CNN's fast interference.

We employ the Raspberry Pi 4, which features a CPU comprising 4xARM A72 cores. In our experiments, we have downclocked the Raspberry Pi 4 to a clock frequency of 1GHz to ensure stability during long benchmark runs.

The DPU runs at a clock frequency of 550 MHz. The synthesis results from Vivado are shown in Table III, and show that the limiting resource for our design is the URAM - our design consumes all 64 available URAMs.

All the platforms used for benchmarking are described in Table II. We have used Raspberry Pi versions 3B+ and 4, a NVIDIA Jetson Nano, a Dell Server using Intel Xeon X3440 CPU and a laptop using the Intel Core i7-4710HQ CPU.

The performance and accuracy results are presented Table IV. As illustrated in Figure 4, our proposed design runs at 43.5 FPS, outpeforming all the other platforms we have used for benchmarking. The experimental results show that our solution is 6.1X faster than the Maxwell GPU on the NVIDIA Jetson Nano. Furthermore, our design maintains similar accuracy as the other platforms tested as shown in Table IV.

VI. IMPACT AND REUSABILITY

The development of this AI-augmented barcode reader project using the Xilinx Kria KV260 may yield substantial

TABLE IV
EXPERIMENTAL RESULTS ON PERFORMANCE AND ACCURACY.

Platform	Processor	Performance [FPS]	Detection accuracy [%]
Raspberry Pi 3B+	CPU 4xARM A53@1.4GHz	0.2943	0.9687
Raspberry Pi 4	CPU 4xARM A72@1GHz	1.9350	0.9190
NVidia Jetson Nano	GPU Maxwell 128 CUDA cores	7.1047	0.9217
NVidia Jeston Nano	CPU 4xARM A57@1.43GHz	1.75	0.9128
KV260	Xilinx DPU@550MHz	43.5	0.9450
KV260	CPU 4xARM A53@1.5GHz	0.91	0.9200
Workstation Dell Poweredge...	CPU Xeon X3440	4.7	0.9100
Laptop	CPU Core i7-4710HQ	16.9	0.9100

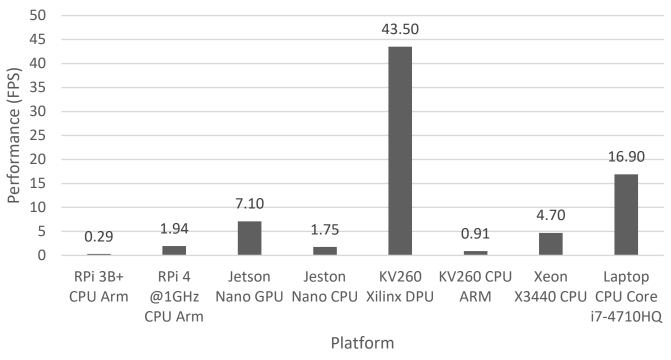

Fig. 4. Performance Results

impact in various domains, for example automated cashiers at local supermarkets who may not afford financial resources for other popular self-scan infrastructure. This innovative solution has introduced several noteworthy implications and benefits:

A. Enhanced Efficiency

Our AI-based application optimizes the process of barcode detection and tracking in retail settings. By automating these tasks, we eliminate manual input, mitigating the risk of human error and significantly enhancing overall operational efficiency. This results in expedited checkout processes and improved customer experiences.

B. Cost-Effectiveness

Our barcode reader offers a cost-effective alternative to existing solutions such as RFID tags and camera arrays. Leveraging the power of AI and FPGA technology, we achieve high-performance barcode detection without the need for expensive hardware or extensive infrastructure. This cost-efficient nature ensures scalability and accessibility for various retail establishments. A self checkout cashier costs approximately 30000$ MSRP according to this article [13]. A Kria KV260 with the basic accesories pack costs approximately $300 MSRP [14].

C. Reusability and Adaptability

The configuration of the Xilinx Kria KV260 and the seamless integration of AI models facilitate the reusability of our system in different scenarios. The modular design of the Xilinx Kria platform allows for effortless customization and adaptation to specific retail environments. Furthermore, the neural network for barcode detection can be retrained or fine-tuned for other applications, offering versatility and flexibility.

D. Potential Environmental Impact

Our solution presents a more sustainable approach compared to RFID tags. Widespread adoption of RFID tags leads to increased production of integrated circuits, which has significant environmental and sustainability implications. By utilizing AI and camera-based barcode detection, we minimize resource consumption and reduce pollution associated with conventional solutions.

VII. RELATED WORK

The field of barcode localization in digital images has been an active area of research for the past decade. Numerous approaches have been proposed, each utilizing different image processing techniques. In this section, we provide an overview of previous works that are relevant to the barcode localization algorithm presented in this paper.

1D barcode localization methods have employed various techniques such as simple image filters, orientation histograms, line detection, morphology operators, Gabor filters, and harmonic analysis [15]–[23]. These methods focus on detecting the edges present in 1D barcodes.

Most existing barcode localization algorithms make certain assumptions about code orientation, scale, or symbology, that limit their applicability. Additionally, these algorithms often struggle with defocused or motion-blurred images, as blur distorts the barcode structures. Recently, a combined barcode localization algorithm that addresses orientation, scale, symbology, and blur invariance has been proposed [17]. This algorithm achieves real-time performance on a smartphone's CPU. However, to further enhance the speed and performance, this paper focuses on implementing the algorithm on the embedded graphics hardware of the Xilinx Kria KV260 using fragment shaders. The existing literature on barcode detection, embedded systems, FPGA-based solutions, and the integration of AI models on the Xilinx Kria KV260 provides valuable insights. Building upon this knowledge, our project aims to develop an AI-enhanced barcode detection system on the Xilinx Kria KV260 platform, incorporating the latest advancements in the field.

VIII. Conclusions and future work

In conclusion, our project presents a versatile solution for an AI-augmented barcode reader using the Xilinx Kria KV260 platform. This solution offers efficient and cost-effective barcode detection and tracking in retail environments. By leveraging AI and FPGA technology, we achieve streamlined checkout processes, scalability, and sustainability advantages. Our solution reduces manual input errors, enables customization, and minimizes environmental impact compared to RFID tags. The project employs computer vision and AI algorithms on a cost effective FPGA-based solution. Our experiments show that our proposed design using the Xilinx Kria KV260 DPU is 6.1X faster than a NVIDIA Jetson Nano GPU.

Overall, our AI-augmented barcode reader has the potential to enhance efficiency and customer experience while paving the way for broader implementation in various domains. and

References

[1] A. Pușcașu, "Ai-augumented barcode reader using xilinx kria kv260," https://github.com/alex2kameboss/xohw23-141.

[2] https://semiwiki.com/semiconductorservices/297821-semiconductors-the-e-waste-problem-and-a-pathway-for-asolution/., accessed: 2023-3-19.

[3] Y. Xiao and Z. Ming, "1d barcode detection via integrated deep-learning and geometric approach," *Applied Sciences*, vol. 9, no. 16, 2019. [Online]. Available: https://www.mdpi.com/2076-3417/9/16/3268

[4] P. Adarsh, P. Rathi, and M. Kumar, "Yolo v3-tiny: Object detection and recognition using one stage improved model," in *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2020, pp. 687–694.

[5] A. Zharkov, A. Vavilin, and I. Zagaynov, "New benchmarks for barcode detection using both synthetic and real data," in *Document Analysis Systems*, X. Bai, D. Karatzas, and D. Lopresti, Eds. Cham: Springer International Publishing, 2020, pp. 481–493.

[6] xiuqizheng@outlook.com, "Barcode detection dataset," https://universe.roboflow.com/xiuqizheng-outlook-com/barcode-detection-buaq8 , apr 2022, visited on 2023-05-25. [Online]. Available: https://universe.roboflow.com/xiuqizheng-outlook-com/barcode-detection-buaq8

[7] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," in *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3464–3468.

[8] "Mot challenge 2017 results," https://motchallenge.net/results/MOT17/, accesat: 20.12.2022.

[9] Xilinx, "Kria kv260 design overview," xilinx.github.io/kria-apps-docs/kv260/2022.1/build/html/docs/smartcamera/docs/introduction.html, accessed: 2023-06-29.

[10] ——, "Kria kv260 vision ai starter kit user guide," https://docs.xilinx.com/r/en-US/ug1089-kv260-starter-kit/Summary, accessed: 2023-06-29.

[11] ——, "Kria k26 som data sheet (ds987)," https://docs.xilinx.com/r/en-US/ds987-k26-som/Overview, accessed: 2023-06-29.

[12] AMD, "Kria k26 system-on-module," https://www.xilinx.com/products/som/kria/k26c-commercial.html, accessed: 2023-06-29.

[13] "Kiosks for grocery store self-check-out," https://www.kompareit.com/business/kiosks-grocery-store-self-check-out.html, accessed: June 20, 2023.

[14] "Xilinx kria kv260 vision starter kit," https://www.xilinx.com/products/som/kria/kv260-vision-starter-kit.html, accessed: June 20, 2023.

[15] Z. Bai, Y. Chen, Z. Yang, and J. Wu, "Simultaneous real-time segmentation of diversified barcode symbols in complex background," in *Proc. First International Conference on Intelligent Networks and Intelligent Systems, 2008, ICINIS'08*, 2008, pp. 527–530.

[16] O. Gallo and R. Manduchi, "Reading 1d barcodes with mobile phones using deformable templates," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 9, pp. 1834–1843, 2011.

[17] G. Sörös and C. Flörkemeier, "Blur-resistant joint 1d and 2d barcode localization for smartphones," in *Proceedings of the 12th International Conference on Mobile and Ubiquitous Multimedia*, ser. MUM '13. New York, NY, USA: Association for Computing Machinery, 2013. [Online]. Available: https://doi.org/10.1145/2541831.2541844

[18] E. Tekin and J. Coughlan, "Blade: Barcode localization and decoding engine," The Smith-Kettlewell Eye Research Institute, Tech. Rep. 2012-RERC.01, December 2012.

[19] I. Szentandrási, A. Herout, and M. Dubská, "Fast detection and recognition of qr codes in high-resolution images," in *Proceedings of the 28th Spring Conference on Computer Graphics*, ser. SCCG '12. New York, NY, USA: Association for Computing Machinery, 2012, p. 129–136. [Online]. Available: https://doi.org/10.1145/2448531.2448548

[20] D. Chai and F. Hock, "Locating and decoding ean-13 barcodes from images captured by digital cameras," in *2005 5th International Conference on Information Communications & Signal Processing*, 2005, pp. 1595–1599.

[21] M. Katona and L. G. Nyul, "Efficient 1d and 2d barcode detection using mathematical morphology," in *Proc. International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing, 2013, ISMM'13*, 2013, pp. 464–475.

[22] M. Wang, L.-N. Li, and Z.-X. Yang, "Gabor filtering-based scale and rotation invariance feature for 2d barcode region detection," in *2010 International Conference on Computer Application and System Modeling (ICCASM 2010)*, vol. 5, 2010, pp. V5–34–V5–37.

[23] A. Tropf and D. Chai, "Locating 1-d bar codes in dct-domain," in *2006 IEEE International Conference on Acoustics Speech and Signal Processing Proceedings*, vol. 2, 2006, pp. II–II.