*Author:*
**Liu, Yanan**

*Title:*
**On-Sensor Visual Inference with A Pixel Processor Array**

# On-Sensor Visual Inference with A Pixel Processor Array

By

YANAN LIU

Bristol Robotics Laboratory & Visual Information Laboratory
UNIVERSITY OF BRISTOL

A dissertation submitted to the University of Bristol in accordance with the requirements of the degree of DOCTOR OF PHILOSOPHY in the Faculty of Engineering.

DECEMBER 2021

Word count: 30,318

# Abstract

On-sensor visual information inference is an emerging technology for the advancement of efficient information extraction which only happens on the sensing chip of the sensory system, providing a means of signal collection, storage, and digestion. Vision is the primary sense of a human being, accounting for a majority of the world's information collection. Although numerous progresses in computer vision have been built upon conventional camera vision systems with their associated methodologies, there are still many constraints, such as system latency and power consumption. Latency introduced by image digitisation, storage, and transmission is a bottleneck that inhibits the conventional machine vision system from responding quickly to changes in their environment. Furthermore, traditional machine vision systems with loose integration of separate sensors and processors have high energy costs, weight and size, making them unsuitable for portable tasks.

Considering these limitations, this thesis thus investigates a new visual information process scheme with an emerging visual sensor: the Pixel Processor Array (PPA) by directly processing signals where they are collected, hence avoiding the aforementioned issues with conventional machine vision systems. In particular, this thesis establishes mobile robotic control systems with on-sensor computed results for multiple navigation research. Then, our work investigates novel parallel visual inference approaches, with a particular emphasis on parallel machine vision algorithms and cutting-edge machine learning-based algorithms. Specifically, we are motivated to perform neural networks to extract higher-level helpful information from the analogue signals. An edge computing platform can be established based on our neural network with the PPA, where only a small quantity of extracted information is obtained, allowing for more efficient data transmission with less bandwidth. Hence, this thesis presents a lightweight and high-speed binary convolutional neural network on the sensor to categorise a range of objects. With the proposed methods to implement networks, all floating-point time-consuming multiplication operations can be replaced by efficient addition/subtraction and bit shifting operations. The focal-plane visual inference is difficult due to hardware resource constraints, such as limited registers and analogue noises. Hence, this work further proposes the purely binarised convolutional neural networks with both binary weights and activations. This thesis trains and implements neural networks with batch normalisation and adaptive threshold to binarise activations. The binary activations on the sensor benefit the neural network performance by alleviating the noises introduced by using analogue signals.

Furthermore, prior research has explored visual sensors for information collection but not for signal processing or motion control; however, this thesis investigates the direction of image processing on the sensing chip and servo motor control using the sensor's digested data directly. Thus, by merging on-sensor neural network inference and direct servo motor control, a sensory-motor system is presented. Moreover, with our proposed dynamic model swapping scheme, more sophisticated classification tasks than earlier work can be achieved. Lastly, a new on-sensor neural network architecture, fully convolutional neural networks, is presented for localisation and coarse segmentation tasks without using the fully connected layers. To deploy this new architecture of a three-layer neural network on the sensor, group convolution is introduced and implemented, with both binary weights and activations, making the fully convolutional neural network compact enough to be embedded on the sensor.

# DEDICATION AND ACKNOWLEDGEMENTS

Firstly, I would like to thank my PhD supervisors, Prof. Walterio Mayol-Cuevas, Prof. Thomas Richardson, and Dr. Laurie Bose, for their advice and support throughout my time at the University of Bristol. In particular, I want to express my gratitude to Walterio and Laurie, who took the initiative and directed me into the brand-new field of on-sensor image processing and artificial intelligence.

I would like to express my gratitude to my parents for their unwavering and unconditional love and support.

Then, I would like to express my gratitude to Prof. Piotr Dudek, Dr. Jianing Chen, and Dr. Steven J. Carey from the University of Manchester for their assistance with a variety of issues in learning and programming the SCAMP Pixel Processor Array and for reviewing my papers especially during COVID-19.

I also would like to thank the following kind individuals: Prof. Ranger Rui Fan, Dr. Alexander W. Pang, Dr. Euan John Kenneth Judd, Dr. Colin Greatwood, Prof. Jianglong Guo, Dr. Chaoqun Xiang, Dr. Yao Lu, Pengcheng Hao, Dr. Will Price, Jian Ma, Xingrui Yang, Yuhang Ming, Xinyu Yang, Tianqi Yang, Zoe Mengxuan Zou, Faegheh, Dr. Xiaoyang Wang, and Simon Chengxi Zeng for their moral support. I will remember these memories while we chatting, thinking, laughing, grumbling, hiking, working out, badminton, jogging, supper, and sightseeing. A special thanks goes out to Dr. Hepeng Ni, an old friend who has inspired me to get out of my comfort zone and view the big picture of my entire professional life through his words of support and advice.

Lastly, I would like to express my gratitude to the UK's EPSRC CDT programme for financing this research especially during the lockdown resulted from COVID-19. To be more precise, this work was financially supported by UK EPSRC under grant number EP/M019454/1, EP/M019284/1 (EPSRC Project Agile[1]), EPSRC Centre for Doctoral Training in Future Autonomous and Robotic Systems: FARSCOPE[2] and China Scholarship Council (No. 201700260083).

[1]https://sites.google.com/view/project-agile/people?authuser=0
[2]https://www.farscope.bris.ac.uk/yanan-liu

To all the unforgettable and beautiful moments experienced in the UK between 9th September 2017 and 15th December 2021.

# AUTHOR'S DECLARATION

I declare that the work in this dissertation was carried out in accordance with the requirements of the University's Regulations and Code of Practice for Research Degree Programmes and that it has not been submitted for any other academic award. Except where indicated by specific reference in the text, the work is the candidate's own work. Work done in collaboration with, or with the assistance of, others, is indicated as such. Any views expressed in the dissertation are those of the author.

SIGNED: .................................................... DATE: ..........................................

# Table of Contents

Page

# List of Tables

# List of Figures

# Chapter 1

# Introduction

This chapter provides an overview of and introduction to this thesis which mainly includes the rationale of the study, on-sensor computing, focal-plane sensor-processor chips [8], the SCAMP vision system [9] and its PPA [10], a literature review, related preliminary knowledge, and the study's contributions to knowledge.

## 1.1 Motivation

Vision is one of the primary sensing modalities through which autonomous systems can interpret their surroundings. However, real-time visual information processing is complex. In great part due to the large amounts of data are involved and at the speed required for fast reacting systems or moving robots, particularly where low mass, low power consumption and cost of the system are of concern. The reasons for these concerns are primarily related to the way data move through the system which is dictated by the highly simplistic and idealised manner in which visual processing is currently performed. In the traditional approach of visual processing, the massive amounts of data that a sensor captures are sent verbatim from image capture to further deep parts, such as GPU/CPUs (Figure 1.1). This only results in substantive inefficiencies as irrelevant data for the task at hand are ferried through the entire system, which limits the latency and power dissipation of the system. To overcome these issues, we need to move data processing nearer the time of image capture. The role of the vision sensor in this approach would no longer be simply to acquire the visual signal, but to digest it, producing meaningful, highly compressed information, instead of video frames. This approach to efficient vision that processes on the image plane inspires our work on visual algorithms for SCAMP Pixel Processor Arrays (PPAs, Figure 1.2).

This thesis explores the on-sensor visual signal processing with proposed machine vision and neural network methods. Specifically, this work presents parallel machine vision algorithms and convolutional neural network (CNN) inference on the focal plane of an emerging and unconventional image sensor: the SCAMP Pixel Processor Array [10–12] (Figure 1.1, 1.2) across various algorithms and applications, covering feature extraction, robot navigation, neural network inference for classification, localisation and segmentation in real-time.

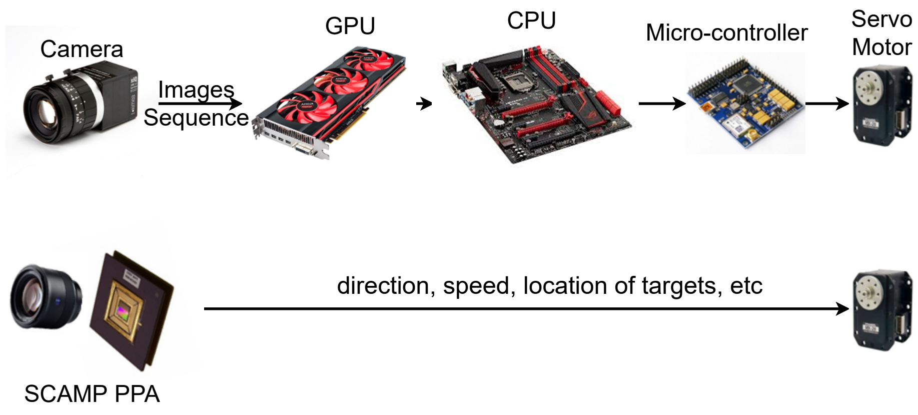


Figure 1.1: A comparison of the workflow between the conventional machine vision system (top) and the SCAMP on-sensor visual computing system (bottom). With meaningful control instructions directly computed from the SCAMP PPA, a significant reduction in external computing units required for image processing can be seen for a vision-based control scheme.

### 1.1.1 Issues with the Conventional Machine Vision Systems

As shown in Figure 1.1, a common conventional machine vision system for motion control usually consists of a camera for image capturing, a CPU/GPU-based image processing system for decision-making, other wired/wireless accessories for data transmission, and actuation motors. This system is widely used in the area of machine vision and robotics [13] based on either conventional image processing or machine learning algorithms [14]. However, this scheme is not suitable for embedded and portable devices and their applications because of its inefficiency in image capturing and transmission, its dependency on the CPU/GPU for both training and inference, high-power consumption (> 50 W) with external CPU/GPU processing units, and heavy load with many components within the machine vision system.

Although there is some work trying to implement image processing and neural networks as close as to the sensor chip [15] or use specific acceleration module [16] on the embedded devices to reduce the device volume, improve the efficiency, and decrease the power consumption, the fundamental machine vision architecture remains similar to Figure 1.1 (top). As far as we are aware, few works deploy neural networks wholly on the sensor, which might result from the lack of suitable hardware platforms or deployment algorithms. This thesis, however, tries to explore a novel visual information process area: image processing directly using the analogue signals with parallelised image algorithms and neural networks, which is to say 'process signals where they are collected'. Thus this thesis takes advantage of the SCAMP-5d vision system as the platform to perform the on-sensor analogue information processing in low-level image feature extraction and high-level neural network inference. To embed image processing and neural networks onto the sensor and then facilitate applications on mobile platforms and portable devices, novel image processing algorithms and methods for signal process on the emerging PPA sensor were developed.

## 1.2 On-Sensor Computing

### 1.2.1 Background

The objective of on-sensor (same as 'in-sensor' as used in other papers [15, 17]) computing is to sense, extract, analyse, store, and compute sensory signals in an in-situ manner within sensors. Unlike the conventional digital electronics-based sensor technology that mainly focuses on data collection with data processing externally, on-sensor signal processing emphasises sensing, signal storage and pre-/process where the analogue signal is collected. By the integration of the sensing, storage, and computing on the sensing plane of the sensory chip, low-power and efficient Edge Computing is enabled for the embedded system, which is meaningful in the area of Internet of Things (IoT) without needing to send redundant information to the cloud, hence reduce the pressure on central computation and data transmission bandwidth.

The idea of on-sensor visual computing is bio-inspired by the mammalian brain and visual system, where the retina pre-processes the visual information and then

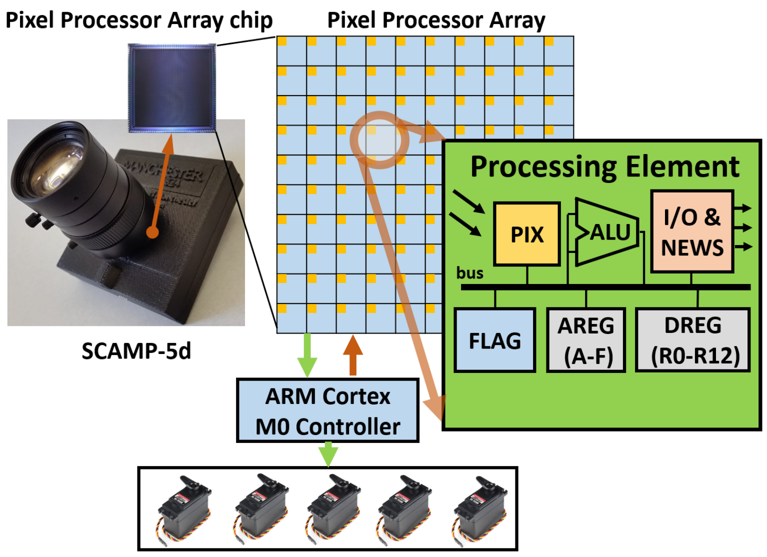


Figure 1.2: SCAMP-5d vision system and its pixel processor array chip. SCAMP-5d consists of PPA with 256×256 Processing Elements (PEs) and ARM Micro-controller where parallel image processing is conducted on PPA by directly operating on analogue signal (electric current from PIX, which is proportional to the light intensity) within Analogue Register (AREG) and bit operation within in Digital Register (DREG). Hence, there is no need for time-consuming and energy-inefficient analogue-to-digital conversion. The PPA is designed to have highly interconnected PE and registers where information can be shared and accessed adjacently, enabling efficient parallel machine vision computing. ARM micro-controller is in charge of sending instructions to the PPA, receiving the processed information from the PPA, extra post-image processing, and more fully-connected layers for deeper CNN.

sends extracted signals to the brain through optical nerves [18]. Nowadays, this on-sensor visual computing technology can be performed in many emerging hardware systems based on the advanced large-scale circuit design, such as SCAMP vision system, photodiode array [19, 20], event camera [21], and memristor [22–24]. Considering the practicability and availability of the hardware platform, this thesis uses the SCAMP-5d vision system as the platform to validate the proposed ideas, methods, and experiments exploring the possibilities of on-sensor visual inference for various applications.

### 1.2.2 On/Near-Sensor Computing Devices

A sensor is used to collect the information relating to the environment in terms of vision, tactics, and olfaction. and send this information to a computer processor after digitisation. Hence, conventional sensors mainly play the role of information collectors.

Table 1.1: List of on/near-sensor processing vision sensors.

| Names | resolution | on/near sensor process | on-sensor storage | speed (FPS) |
|---|---|---|---|---|
| SONY IMX500 [25] | 4056×3040 | near-sensor (Pixel+DSP) | RGB | 240 (1080p) |
| Aistorm Mantis2 [26] | 96×96 | on sensor | gray-scale image | 50K |
| Eye-RIS [1] | 176×144 | on-sensor | 7 gray + 4 binary | 10K |
| Memristor [27] | no sensing | on-sensor | – | – |
| Photodiode Array[19] | on-sensor | on-sensor | – | 20M bins |
| SCAMP | 256×256 | on-sensor | 7 gray + 13 binary | 100K [10] |
| DVS [21] | – | on-sensor pre-processing | binary events | >10K |

In recent years, with the development of techniques on integrated circuit design and the edge-computing requirements, a sensor is gradually integrated with the ability of signal processing, which is different from general-purpose computers. Before entirely on-sensor computing, near-sensor computing was introduced to make the visual sensors more compact by placing the processor closer to the pixels. This subsection introduces several emerging and unconventional on/near-sensor visual computing sensors to show the recent research development in on/near sensor computing. A comparison among them can be seen in Table 1.1.

**SONY IMX500** [25]: It is the so-called the first edge-processing intelligent vision sensor. SONY IMX500 stacks an upper pixel layer and a lower logic layer. The pixel layer, like normal CCD/CMOS sensors, takes charge of image capturing. The logic layer consists of a DSP (Digital Signal Processor) and dedicated on-chip SRAM (Static Random-Access Memory) for processing and storage, respectively. Artificial intelligence functions, such as CNN, are embedded in the logic layer to process the pixel layer signal directly. With this two-layer architecture of sensing and processing, only metadata results can be read out for applications, including image classification, object detection, pose detection, and semantic image segmentation. However, strictly speaking, SONY IMX500 is near-sensor computing rather than on-sensor computing since the image should be captured first, then digitised and transferred onto the DSP layer for processing.

**Aistorm Mantis2** [26]: Mantis system is based on the event-driven charge domain for analogue signal processing without digitisation and provides an 'always on' solution for analogue signal processing. One of the main features claimed by Aistorm is the noise cancelling techniques associated with the analogue signal. In addition, artificial intelligence can also be integrated into a chip for many applications. However, the latest Mantis product only has a resolution of 96×96, which is challenging to tasks that normally require a higher resolution.

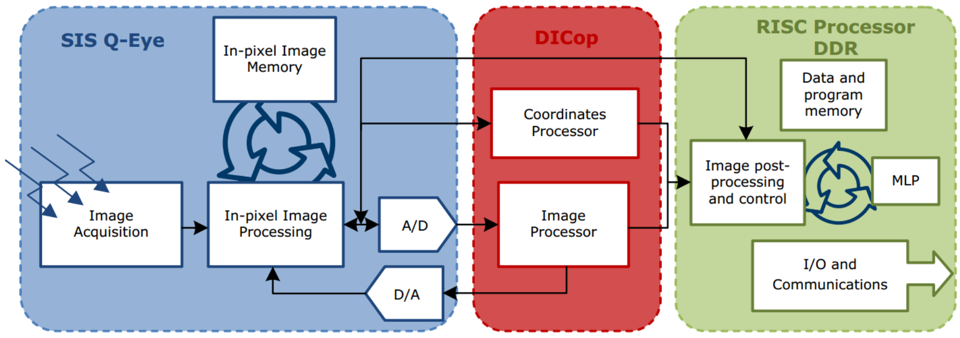


Figure 1.3: Eye-RIS vision system function diagram. Figure from [1].

**SCAMP Vision System:** SCAMP Vision Sensors [10] (Figure 1.2) include a massively parallel Single Instruction, Multiple Data (SIMD) processor array within the image sensor device's pixels. In contrast to a traditional image sensor, it does not output raw pictures but rather the results of on-sensor calculations, such as a feature map, optic flow map, and/or address-events specifying the positions of the pixels of interest. The sensor is programmable, allowing for the execution of a wide range of vision algorithms. Due to the fact that early vision calculations are performed fully on-sensor, the whole camera system operates at a fast rate and consumes little power, allowing novel embedded vision applications in fields, such as robotics, virtual reality, automotive, and surveillance. See section 1.3 for more details. Notice that other names for a similar type of focal-plane sensor processor can be seen from [8] with names, e.g. CPA, FPSP.

SCAMP vision systems are available in a variety of configurations at the moment (Figure 1.7). The proposed algorithms in this thesis are based on SCAMP-5, which can also be easily extended to SCAMP-7 with the similar amount of registers and same resolution. As for the previous version of the SCAMP, the implementation of the algorithms should be slightly adjusted to accommodate the available hardware resources. In addition, due to the lack of access, our proposed methods in this thesis are not implemented on other on-sensor computing platforms yet. However, given the similar hardware design, it is promising to deploy our algorithms on Eye-RIS using the proper implementation techniques and libraries offered by the Eye-RIS development environment. Because of the different hardware design of other vision sensors listed in Table 1.1, it would be challenging to expand our methods to those systems.

**Eye-RIS:** [1] Eye-RIS commercial vision system on Chip (VSoC) extends CMOS pixel functionality with image storage (7 gray-scale images and 4 binary images) and digital/analogue signal processing ability. Specifically, a 32-bit RISC (Reduced Instruction

Set Computer) is integrated with a vision sensor for image post-processing after the parallel on-sensor pre-processing. The resolution of Eye-RIS vision sensor is 176×144. Notice that Eye-RIS's overall functional diagram (Figure 1.3) is similar to that in the SCAMP vision system, where the counterpart of RISC is M0 micro-controller in the SCAMP PPA. However, the most significant difference is that the Eye-RIS contains a DICop part, a digital image coprocessor dealing with geometric transforms whose results can be sent back to the pixel level for further processing.

**Memristor-Related Devices:** [27] Memristor-based hardware is a platform to deploy the neural network using the programmable resistance within the integrated circuits mimicking the synaptic connections in a human brain [22–24, 27, 28]. However, it integrates only storage and processing function. Hence signals should be input from sensors or other storage devices. They are thus usually integrated with other sensory systems for information processing.

**Dynamic Vision Sensor (DVS):** [21] DVS produces data in the form of sparse contrast-change events that facilitate low-latency visual processing using external computational hardware [29–31]. These binary events are generated from on-sensor processing according to the brightness changes. Although the pixels in a DVS have a primitive on-sensor processing ability by binarising brightness changes, it achieves an ultra-high speed response to the environment when working with external hardware computing units, enabling a huge potential for robotics and computer vision in a challenging environment [32].

**Other Emerging Sensor Devices:** Mennel *et al.* [19] use a 2D semiconductor ($WSe_2$) photodiode array as the vision sensor, as well as the photoresponsivity matrix to store the connecting weights of the neural network, where both supervised and unsupervised learning for classification is present. However, laser light and optical systems are needed to project images onto the chip, preventing regular use. Song *et al.* [20] proposed a CMOS-based PIP (Processing-in-Pixel) architecture where image convolution (8-bit weight configuration) can be performed as the image pre-processing before image data is readout.

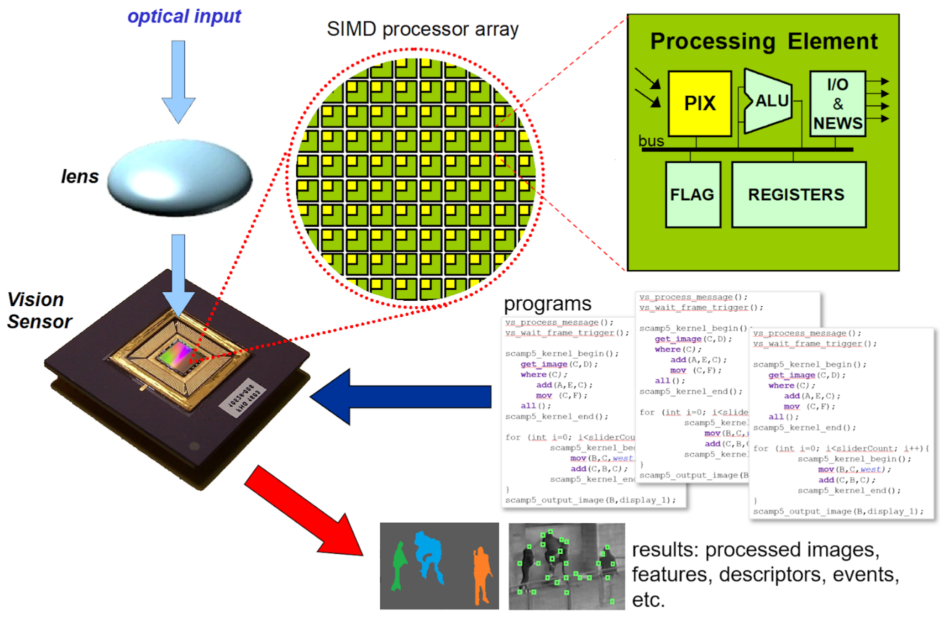
## 1.3 The Pixel Processor Array and SCAMP Vision System

As the name implies, the concept of Pixel Processor Array (PPA) integrates each pixel with its own processor and then organised in a 2D array of a chip plane. The SCAMP vision system (Figure. 1.2) [33–37] is a representative example of the PPA which has been invented, designed, and developed by the University of Manchester since the 1990s. The motivation is to design a fully programmable general-purpose SIMD cellular processor array avoiding analogue-to-digital conversion (ADC) by performing processing directly with analogue signal for a novel world-machine interface that can sense, store, and reason without relying on the external centralised processing units [11, 38].

In terms of hardware techniques, the core sensing and computing chip of the SCAMP vision system, the PPA, integrates information storage on registers, image processing and analogue information operation (arithmetic operation, shifting, *et al*, digital/bit operation, and logical operations). As can be seen from Figure 1.2, as for hardware resources, for each Processing Element (PE), there are seven '8-bit' read/write AREG (A to F) which can be used for signed value storage and computation with basic arithmetic operations, such as addition, subtraction, division (1/2), etc. In addition, thirteen 1-bit DREG (R0 to R12) in each PE (256×256 in total) can execute the Boolean logical operations, such as AND, OR, XNOR, and NOT [39] with information after binary thresholding on AREG. Each register in PE executes identical instructions synchronously, hence enabling parallel image processing. In addition, the FLAG register can activate different areas of registers given corresponding patterns for more flexible operation. With the neighbour access function where each pixel is able to communicate with its four neighbours (north, west, east, south), an efficient parallel image shifting can be implemented easily. The ARM-based micro-controller dispatches instructions to the PPA with a Cortex M0 running at 204 MHz. The analogue operations are executed at 5 MHz and digital at 10 MHz. Other I/O functions, such as USB2.0, GPIO, SPI, and UART, are performed on Cortex M4 Core [39]. Figure 1.4 shows the development architecture with SCAMP vision system, where instructions (programmed using C++ language) of the PPA are allocated by the ARM micro-controller [39].

## 1.4 On-Sensor Machine Vision with the Pixel Processor Array

This section explains the emerging notion of on-sensor computing and reviews the research and applications on the machine vision using the SCAMP PPA sensor, which provides preliminary knowledge to later chapters on image processing and neural networks with the PPA. This section focuses on reviewing the on-sensor computing algorithms developed on the PPA, hopefully sending inspiration to other researchers and developers



```
vs_process_message();
vs_wait_frame_trigger();

scamp5_kernel_begin();
   get_image(C,D);
   where(C);
      add(A,E,C);
      mov  (C,F);
   all();
scamp5_kernel_end();

for (int i=0; i<sliderCount; i++){
      scamp5_kernel_begin();
         mov(B,C,west);
         add(C,B,C);
      scamp5_kernel_end();
}
scamp5_output_image(B,display_1);
```

Figure 1.4: The SCAMP vision development platform. The processing elements execute software programs, performing image computations on-sensor and outputting high-level information to the rest of the system [12].

who are working with raw analogue sensory data. Figure 1.5 selects some representative studies on the SCAMP PPA since 2006.

### 1.4.1 On-Sensor Analogue Computing with a PPA

On-sensor computing ([15]) is an emerging concept which means carrying out processing on the sensing chip by directly utilising the collected analogue signal (electric current and voltage, for example) without Analogue-Digital Conversion (ADC). On-sensor computing is enabled by an up-to-date circuit design [15]. Nowadays, the data deluge results from ubiquitous sensors may obscure the beneficial information, hence encouraging the edge on-sensor computing device to extract only a small amount of useful information [41]. The PPA is such a device directly operating on the current, enabling focal-plane image processing with the raw sensor data. Notice that on-sensor computing is not limited to the visual information but also can be extended to other forms of a sensor system, such as auditory, olfactory, thermal, and tactile sensor systems [15]. Figure 1.8 shows the ability of the PPA to extract 2D localisation information with a huge superfluous data reduction.

**Efficiency and Low Latency** It can be seen from Figure 1.7 that on-sensor com-

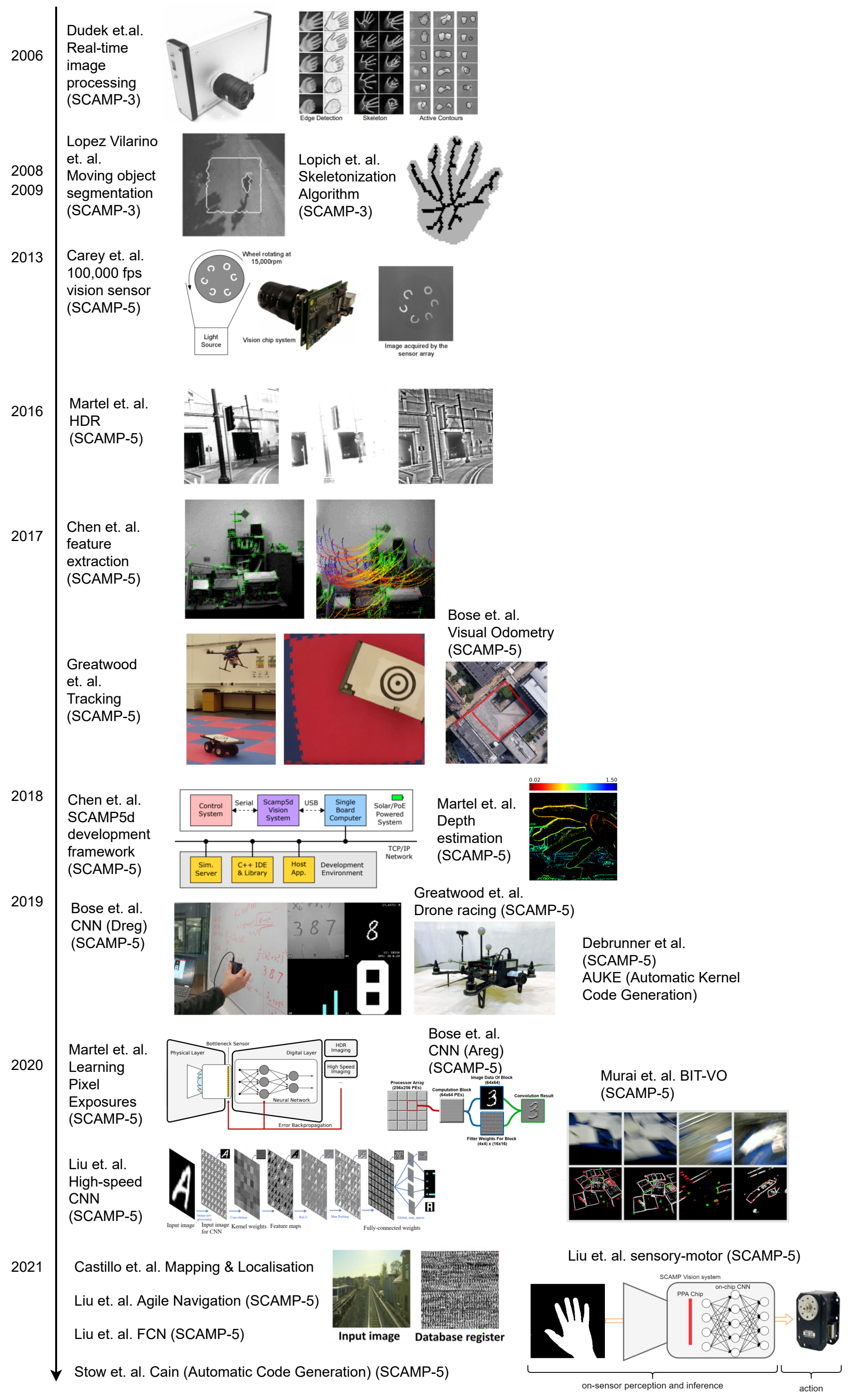

2006
Dudek et.al. Real-time image processing (SCAMP-3)
Edge Detection
Skeleton
Active Contours
2008
2009
Lopez Vilarino et. al. Moving object segmentation (SCAMP-3)
Lopich et. al. Skeletonization Algorithm (SCAMP-3)
2013
Carey et. al. 100,000 fps vision sensor (SCAMP-5)
Wheel rotating at 15,000rpm
Light Source
Vision chip system
Image acquired by the sensor array
2016
Martel et. al. HDR (SCAMP-5)
2017
Chen et. al. feature extraction (SCAMP-5)
Bose et. al. Visual Odometry (SCAMP-5)
Greatwood et. al. Tracking (SCAMP-5)
2018
Chen et. al. SCAMP5d development framework (SCAMP-5)
Control System
Serial
Scamp5d Vision System
USB
Single Board Computer
Solar/PoE Powered System
TCP/IP Network
Sim. Server
C++ IDE & Library
Host App.
Development Environment
Martel et. al. Depth estimation (SCAMP-5)
2019
Bose et. al. CNN (Dreg) (SCAMP-5)
Greatwood et. al. Drone racing (SCAMP-5)
Debrunner et al. (SCAMP-5) AUKE (Automatic Kernel Code Generation)
2020
Martel et. al. Learning Pixel Exposures (SCAMP-5)
Bottleneck Sensor
Physical Layer
Digital Layer
HDR Imaging
High Speed Imaging
Neural Network
Error Backpropagation
Bose et. al. CNN (Areg) (SCAMP-5)
Murai et. al. BIT-VO (SCAMP-5)
Liu et. al. High-speed CNN (SCAMP-5)
Input image
Kernel weights
Feature maps
Fully-connected weights
2021
Castillo et. al. Mapping & Localisation
Liu et. al. Agile Navigation (SCAMP-5)
Liu et. al. FCN (SCAMP-5)
Stow et. al. Cain (Automatic Code Generation) (SCAMP-5)
Input image
Database register
Liu et. al. sensory-motor (SCAMP-5)
SCAMP Vision system
on-chip CNN
PPA Chip
on-sensor perception and inference
action

Figure 1.5: Milestones SCAMP PPA-based work and key SCAMP PPA studies and applications during last 15 years.

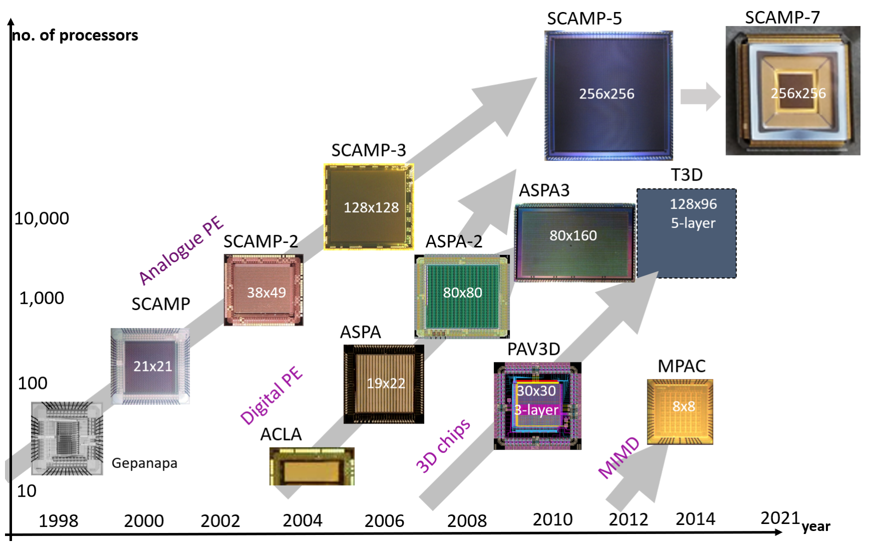


Figure 1.6: The development process of SCAMP series vision system from the University of Manchester. This review mainly focuses on the SCAMP-3 and SCAMP-5 vision system because their higher resolution and performance would enable more research and applications. Source: Piotr Dudek's talk in Second International Workshop on Event-based Vision and Smart Cameras (CVPRW) [40].

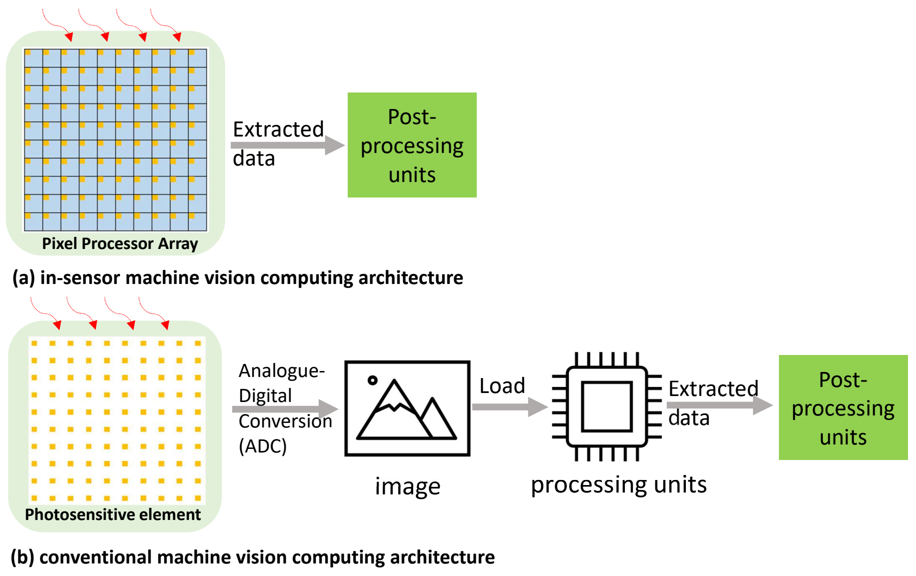

Pixel Processor Array
Extracted data
Post-processing units
(a) in-sensor machine vision computing architecture
Photosensitive element
Analogue-Digital Conversion (ADC)
image
Load
processing units
Extracted data
Post-processing units
(b) conventional machine vision computing architecture

Figure 1.7: Machine vision computing architecture with a SCAMP PPA and a conventional processing device. (a): analogue signal can be directly utilised for machine vision task on the SCAMP PPA. (b): As for the image processing with conventional vision device, light density needs to be firstly read out and converted to digital image which would then be loaded into processing units (GPU/CPU) for useful information extraction.

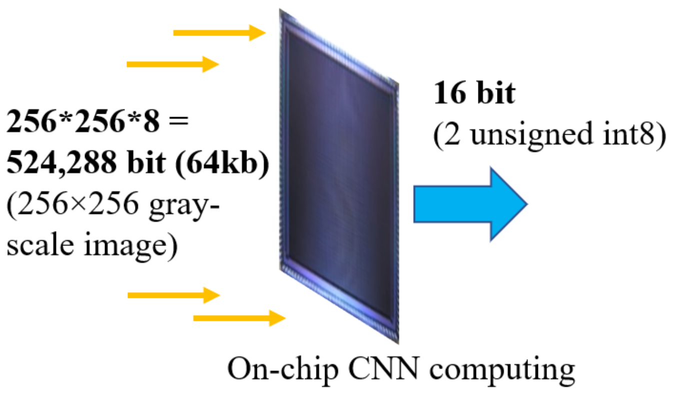


Figure 1.8: The information extraction capabilities with ×32,000 data reduction and a low operation power on PPA using the proposed the FCN for 2D localisation.

puting skips signal digitisation, transmission, and storage process, hence enabling high-speed image processing [10] and CNN inference with low latency and energy cost, which can be integrated with agile, mobile robot platforms [42–45]. In addition, the de-centralised PE distribution and SIMD (single instruction, multiple data) on PEs allow efficient signal processing.

**Low Power Consumption** According to Figure 1.1 and Figure 1.7, there is no external processing units or data process needed. Hence the power consumption can be saved to a large degree. For example, the maximum power cost of the SCAMP-3 vision system for a complex object tracking and counting system is 29 mW [46]. This feature makes SCAMP suitable for mobile platforms, usually with short-battery life.

**Data Security** A unique but non-negligible feature of on-sensor analogue computing is its inherent data security resulting from the direct processing with raw sensory data without recording, transmission, or storage onto external units. Usually, the only output after analogue computing is the high-level target information which is barely reversible to get the original data. Hence, privacy can be strictly protected without source image data leaving the focal plane.

### 1.4.2 On-Sensor Machine Vision with the SCAMP PPA

The algorithms of SCAMP PPA proposed earlier mainly focus on low-level image processing and/or machine vision methods to enhance image quality and extract basic textures with combinations of inherent built-in functions based on SCAMP-3 and SCAMP-5 with a PE resolution of 128×128 and 256×256, respectively (Figure 1.6). It should be noticed that these developed image processing methods are deeply related to the hardware

Table 1.2: List of main studies with the PPA.

| Tasks | Study | Applications |
|---|---|---|
| background extraction | [47, 48] | segmentation |
| contour extraction | [49–51] | object detection |
| skeleton extraction | [52–54] | shape simplification |
| HDR | [49, 55–57] | image enhancement |
| feature corner/edge extraction | [58] | edge/feature based VO [59, 60] |
| target detection/localisation | [10, 61, 62] | high-speed object tracking |
| neural network | [63–68], | high-level inference |
| depth estimation | [67, 69–71] | robot navigation |
| visual odometry | [59, 60] | robot navigation [43, 44] |
| automatic code generation | [72, 73] | neural network inference, face detection |

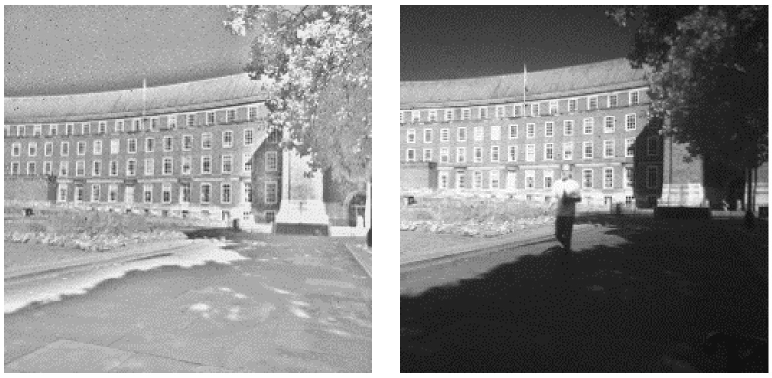


Figure 1.9: Examples of two images with(left)/without(right) HDR algorithms towards the same scene in an outdoor environment.

design of the SCAMP vision system. For example, common methods used in this period are cellular-based algorithms, including cellular neural networks, because the SCAMP PPA itself is a cellular processor array.

#### 1.4.2.1 Image Enhancement

Image enhancement comes along with the imaging process on the PPA compared to the conventional image enhancement which only happens after the image data is captured. Later, other methods are exploited on different image processing tasks. For example, Wang *et al*. [74] proposed a simple coarse grain mapping method to process bigger images than the PPA resolution itself by temporarily storing sub-images into different registers.

HDR (Figure 1.9) is a basic low-level image pre-processing method to obtain rich image information even facing extreme lighting conditions, such as the mixture of dim

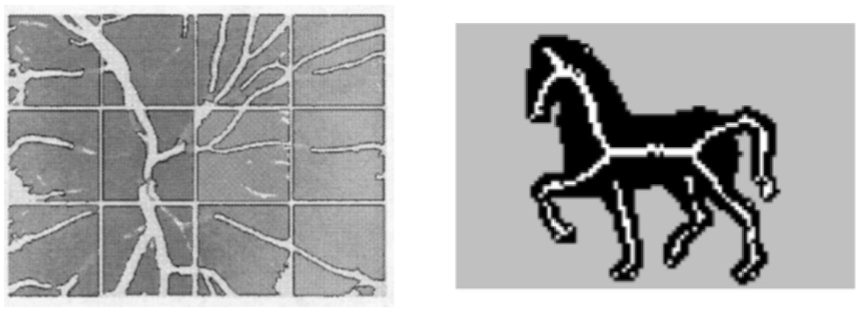


Figure 1.10: Examples of image contour and skeleton extraction using SCAMP PPA. Left: Extracted Retinal Vascular Tree, Figure from [51]. Right: Extracted skeletons, Figure from [48].

and strong light intensity. However, conventional image sensors rely on either a global or rolling shutter to form an image, which limits the efficiency of HDR imaging [57, 75]. Back in 2006, Dudek [49] proposed sensor-level adaptive sensing and image processing with SCAMP-3 [46, 76], where different exposure settings are combined for an image with a high dynamic range. Martel *et al*. [56] make significant contributions in this area using the PPA. The first HDR image generation on-sensor is from [55] where pixel-wise exposure can be controlled to generate HDR images, followed by automotive applications [77]. Furthermore, Bose *et al*., [59] take advantage of the HDR image to extract edges as the robust input information for visual odometry estimation. However, the usage of iterative exposure for different regions of the image slows down the image pre-processing. To speed up the HDR imaging, Martel *et al*. [57] propose the learning shutter function for PEs to expose each pixel independently with an end-to-end training strategy. They obtain an exposure function by training a U-Net neural network and compiling these trained functions on the sensor for inference.

#### 1.4.2.2 Contour and Skeleton Extraction

Contour is an important feature for objects within an image, which can help to identify different entities. Contour extraction algorithms were proposed based on a pixel-level snake with very low latency [50]. In 2007, Alonso-Montes *et al*. proposed the on-sensor automatic retinal vessel tree extraction based on the Cellular Neural Networks [51]. The shared key methods for these work [49–51] is to extract contour iteratively based on the active contour model and Cellular Neural Networks. In 2008, [78] proposed an image pre-processing method based on the cellular automata for a robotic scenario. The skeleton within a binary image shows the object size, position, and simplified

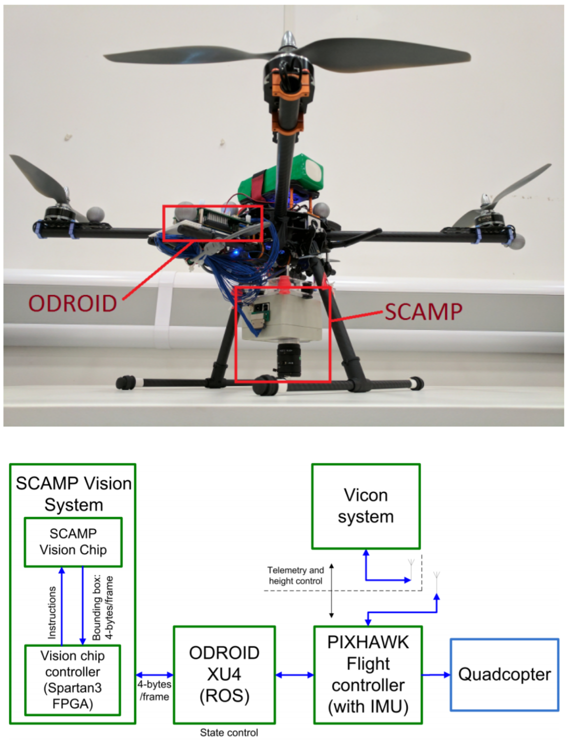

ODROID
SCAMP
SCAMP Vision System
SCAMP Vision Chip
Instructions
Bounding box: 4-bytes/frame
Vision chip controller (Spartan3 FPGA)
4-bytes /frame
ODROID XU4 (ROS)
State control
Vicon system
Telemetry and height control
PIXHAWK Flight controller (with IMU)
Quadcopter

Figure 1.11: Top: The integration of a quadrotor and SCAMP-5 vision system for object tracking. Bottom: a diagram of system hardware (Figure from [42]).

shape. Fast image skeletonization [52] is implemented by [53] based on the wave-trigger propagation/collision. Examples of image contour and skeleton extraction based on the SCAMP PPA can be seen from Figure 1.10.

#### 1.4.2.3 Other Feature Extraction Methods

Other image processing methods, such as background extraction, is exploited by Wang et al. [47, 48]. For higher-level feature extraction, the edge feature can be obtained by deploying Sobel kernel filters or Laplacian filters, which are used in the later work for focal plane visual odometry [59] and neural networks [67]. As for other features, such as corner points extraction, Chen [58] utilised the DREG for corner points extraction based on the FAST16 algorithm, which is used in later work on visual odometry [60]. Based on the above-mentioned low- and mid-level image processing methods, researchers are motivated to exploit more general high-level image processing with up-to-date techniques by taking advantage of the earlier milestone work and the state-of-art progress, such as neural networks.

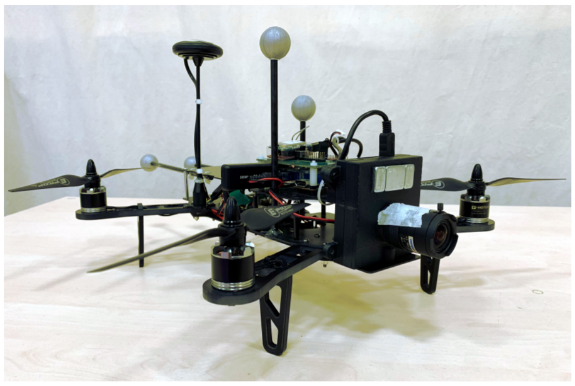


Figure 1.12: Quadrotor setup for the drone racing with a front-facing SCAMP. (Figure from [2])

### 1.4.3 On-Sensor Visual Feature Extraction for Robots

Two major constraints that preclude mobile robots from long-term and diverse applications are their short battery life and limited load. Emerging sensors may hold the key to solving this challenge due to their unique low-level hardware design. The portable SCAMP-5d vision system (171g including the lens) can perform spatial AI processing on-sensor, reducing data transfer pressure between sensing and the main processor, hence increasing overall processing efficiency while maintaining low power consumption [79].

#### 1.4.3.1 SCAMP PPA on a Quadrocopter

The SCAMP-5d vision system has been integrated into quadrocopter systems for target tracking, visual odometry and racing. Greatwood *et al*. perform various experiments by integrating a SCAMP-5d vision system and a quadrotor [2, 42, 43]. Figure 1.11 shows a flight control system in terms of hardware integration and control block diagram, where a pre-set target can be tracked with extracted useful information on sensor even facing short periods of target tracking loss [42]. In this application, the direct on-sensor target position extraction releases the pressure of image capturing, transmission and processing for the whole system. Later, Greatwood *et al*. proposed the on-sensor visual odometry using perspective correction on an agile micro air vehicle based on a similar hardware platform. After that, a drone racing 1.12 within a pre-set environment is demonstrated by taking advantage of the efficient image processing ability on the PPA [2], where the target position can be estimated at around 500 FPS. McConville *et al*. [44] apply the on-sensor visual odometry developed by Bose *et al*. [59] on an unmanned aerial system

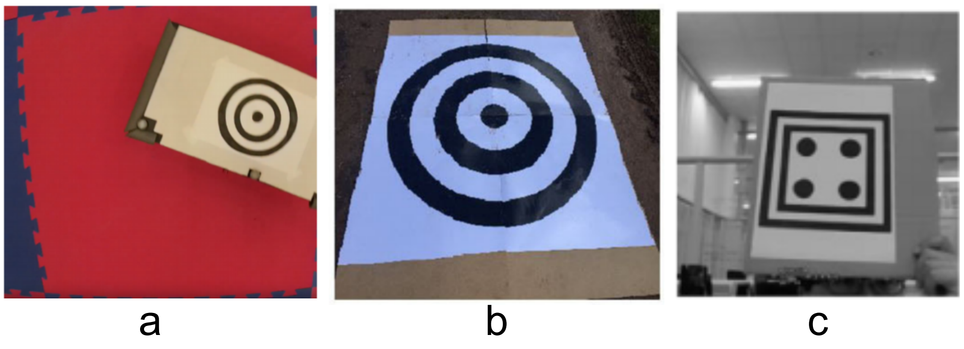


Figure 1.13: Tracking pattern for the drone and ground vehicle. a) Tracking a ground vehicle [42], b) Tracking a moving target while performing a visual odometry [44], c) Tracking a fixed pattern with a mobile ground vehicle [45].

for real-time control purposes.

#### 1.4.3.2 SCAMP PPA for Mobile Robot Reactive Navigation

In terms of navigation with a SCAMP PPA, Liu *et al*. [45] proposed reactive agile navigation on a non-holonomic ground vehicle using PPA by robustly recognising pre-set patterns out of complex environment background. Although very efficient and accurate, using a pre-set fixed pattern for target tracking is difficult to expand in the generalised environment where there are usually random features. With this in mind, Chen *et al*. [67] use in-focal plane feature extraction from the environment to perform a recurrent neural network on the M4 micro-controller using this extracted information to estimate the proximity to the ambient objects for obstacle avoidance purposes. Similar pattern of concentric circles was employed in [42, 44, 45] to effectively extract the dot centre in the circles out of the complex environment.

#### 1.4.3.3 On-Sensor Computing for Mapping and Localisation

Mapping and localisation are useful techniques for robot navigation. On-sensor mapping and localisation is a lightweight and low power cost solution for mobile platforms. Castillo-Elizalde *et al*. [7] for the first time proposed 1-D mapping and localisation by extracting features from a sequence of images as the database first and then localising the incoming image by comparing it with the database and the prior knowledge of the motion model. In their work, two methods were utilised to downsample the original images: direct resizing and local binary pattern to apply them to different localisation situations.

#### 1.4.3.4 Pose and Depth Estimation

For decades, egocentric state estimation has been studied using conventional cameras, emerging DVS devices, and CPU/GPUs. In recent years, there have been some studies utilising SCAMP PPA. For example, Bose *et al.* [59] for the first time, proposed in-sensor 4 Degree-of-Freedom (DoF) visual odometry wholly on the sensor by mapping the real-time input image with the previous keyframe through image scaling, shifting, rotation and alignment. They demonstrate the visual odometry estimation at over 1000 Hz with around 2 W power cost. Debrunner *et al.* [80] use the SCAMP to estimate its global motion with the tiling method at 60 FPS with a low power cost of 100.2 mW. After that, Murai [60] proposed 6 DoF visual odometry based on edge and corner points extracted on sensor and post-processing on a computer with a frame rate of 300 FPS. They take advantage of feature edge, and corner extraction methods [58] and calculate the visual odometry off sensor using a similar strategy with the standard Visual Odometry (VO) systems [81]. Although they combine on-sensor feature extraction and ready-to-use VO computing method off the sensor, it is promising to be a direction in the future to combine the efficient image pre-processing on-sensor and high-volume post-processing with a powerful CPU/GPU, especially when facing storage shortage and general calculation resources for the large-scale computing.

In addition, the SCAMP vision system can also work with other accessories to share the computation burden for more applications. For example, Martel *et al.* [69–71] mounted a controllable liquid lens to generate a semi-dense map in real-time, which is the first work on depth estimation to take advantage of external physical accessories. With this focus-tunable lens, a vast amount of computation pressure on the sensor is relieved. This on-sensor feature extraction and post-image processing on controller scheme are also widely used in many different applications [60, 67], where the task requirement of storage and computing resources is out of the capacity of the PPA.

### 1.4.4 On-Sensor Cellular Automata

The PPA itself is a cellular neural network architecture where each 'cell' is closely connected with its four neighbours, hence information can be shared efficiently. With this in mind, the author is inspired to explore the possibility to perform cellular behaviour, such as Conway's game of life (demonstration shown from[1]) and elementary cellular

[1]https://youtu.be/X_t4c3f-T4s

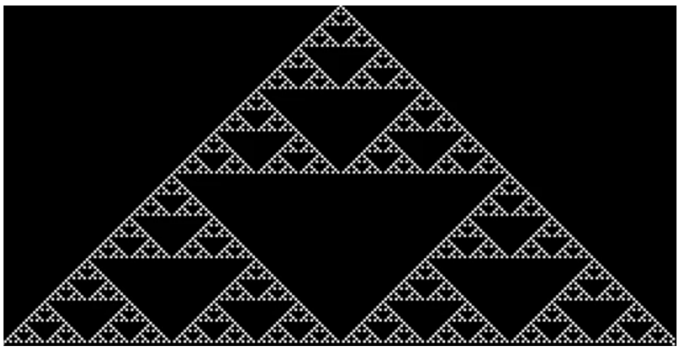


Figure 1.14: Our demonstration of elementary cellular automata with Rule 90 on the SCAMP PPA. This pattern is generated from top to bottom.

automata (demonstration Rule 90 seen from [2]) based on the theory of cellular automata [82]. With the rule of the game of life, all 'cells' can update their states (alive or dead) on-sensor as fast as 53 microseconds for each iteration based on the bit-operation with DREG. As can be seen from Figure 1.14, a Sierpiński triangle is efficiently generated based on bit operation on the sensor with 730 microseconds of 255 iterations to fill the whole chip. In the future, more image processing-related work can be potentially be explored as long as proper update rules and associated steps are trained with neural network methods [83, 84].

## 1.5 On-Sensor Neural Networks with the Pixel Processor Array

### 1.5.1 Neural Network Compression for the Embedded Devices

Different from deep neural networks with millions of float-point weights and biases on the state-of-the-art CPU/GPU, the neural networks for embedded devices should be light-weighted in terms of compression, simplification and quantization, considering the stretched computing and storage resources as illustrated in Section 1.3. This section illustrates the prevalent neural network compression methods with specific hardware platforms.

[2] https://youtu.be/HgPvoK5EJ_s

#### 1.5.1.1 Network Compression Method

Nowadays, many types of neural network compression methods are proposed to accelerate the inference speed and reduce the storage requirement of the deep neural networks. In addition, the deployment range of neural network can be expanded after compression from general-purpose GPU/CPU to specific embedded devices. These compression methods mainly consist of parameter quantisation, pruning, low-rank factorisation, and knowledge distillation [85].

**Parameter Quantisation and Binarisation:** The neural network quantization method converts 32-bit floating-point weights and/or activations into lower-bit parameters, such as 16-bit [86], 8-bit [87], 2-bit (ternary) [88] , and even 1-bit (binary) [89–91] during training process. In the extreme case of neural network binarisation with an idea of learning the binary neuron connections and activations during network training. The typical and popular ones include BinaryConnect [89], XNOR-Net [91], BinaryNet [90] and Ternary Weight Networks [88]. Although the model size and inference efficiency can be significantly optimised for the embedded devices, quantisation's disadvantage, especially for binarisation, is the accuracy gap compared to the normal deep networks. A trade-off should be made to achieve a good balance between accuracy loss and model deployment performance.

**Pruning:** Network pruning [92, 93] reduces the storage requirement of deep neural networks by getting rid of unimportant and redundant connections among neurons. In addition, methods using sparsity regularisation is another way to directly train networks with sparse neuron connections [94]. However, network pruning with regularisation usually more iterations to train to converge with good high-parameter tuning. In addition, neural network pruning decreases the size of network model, but it does not boost the inference efficiency [85].

**Low-Rank Factorisation:** The idea of low-rank factorisation is to approximate the 3D tensor in convolutional layers and fully-connected layers with the decomposed low-rank tensors [95]. In a neural network, many filters that are low-rank matrix which contains redundant information, hence these filters can be decomposed into two matrix with a smaller amount of parameters. However, it is usually computationally intensive to decompose matrix during the training process [85].

**Knowledge Distillation:** Knowledge distillation is another representative compression technique for transferring learned knowledge from a large neural network to a smaller model [96]. To be more precise, a larger capacity 'teacher' neural network can be distilled to train a compact 'student' neural network using its more comprehensive output information (soft target rather than hard target, for example) [97].

In conclusion, different network compression techniques may be more appropriate for specific activities on specific deployment platforms. Given the SCAMP vision system's limited storage capacity and the effectiveness of employing DREG to store binary weights, this thesis makes use of binary neural networks for inference.

## 1.6 Literature Review

This section reviews related work on robotic reactive navigation and neural networks with the emerging visual sensors. We introduced our proposed methods by analysing the connection and difference between existing literature and our work.

### 1.6.1 Robotic Reactive Navigation

Currently, there are lots of studies and applications related to reactive navigation. Tobaruela *et al*. made a thorough analysis on robot navigation by dividing the robot navigation into three different paradigms: deliberative, reactive and hybrid. This classification was based on how the robot processes its sensory information [98]. Then, the reactive paradigm is further divided into the non-purely reactive navigation and purely reactive navigation. The main difference between them is the use of the short memory, in which the latter paradigm only uses the real-time sensor readings, while the non-purely reactive relies on both the current and previous data collected by the sensor. And the deliberative navigation involves the deep sensory information process and follows a scheme of sense-plan-act. However, the reactive navigation paradigm relies on the idea of sense-act. Li *et al*. [99] proposed a visual landmark-recognition system based on reactive navigation technique and developed a fully autonomous mobile robot. Their mobile robot system can generate corresponding motions according to different landmarks. Muhannad *et al*. [100] developed a novel real-time collision avoidance method for mobile robots. Reactive navigation is employed in their application for motion planning and mobile robot control, which is able to detect changes in the environment and dynamically re-routing itself. Brooks [101] claim "the world itself is the best model", which means that all required information and intelligence can originate from the world itself and it is not necessary to build a model of the environment. As a type of reactive navigation scheme, the Braitenberg vehicle [3] is a classic and straightforward task-oriented mobile robot (Figure 1.15) and it is one of the earliest reactive navigation applications. It features

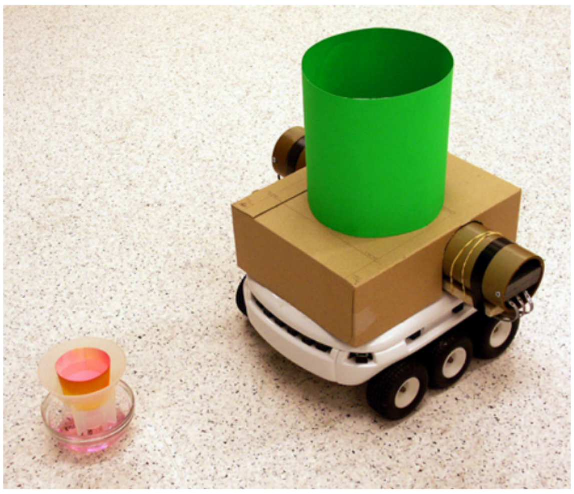


Figure 1.15: The smelling Braitenberg vehicle [3].

the direct link between the wheel motors and the smelling sensor, and the Braitenberg vehicle can reach the odour source without environmental information. In 2011, Petres *et al*. [102] applied the reactive navigation technique on their autonomous sailboat and used the potential field to plan the path. There are studies on landmark and object tracking using reactive navigation methods. For example, Bryan *et al*. [103] utilised the reactive navigation method to track targets and avoid obstacles with a UAV. In their scenario, the visual information extracted from the camera is used to guide the UAV to track an object in 3D space and plan the trajectory in real-time. Although it is a good demonstration of reactive navigation, the UAV in their scenario has to fly slowly to keep track of the target.

Nowadays, one of the common alternatives to traditional cameras is the event camera, such as the Dynamic Vision Sensor (DVS) (Figure 1.16, 1.17). Different from the conventional frame-based visual sensor where the values of pixels are sampled repetitively even if the images remain unchanged, the DVS is only sensitive to changes in intensity, and each pixel responds asynchronously according to the change of its pixel value [104]. A DVS address event is emitted when there is a luminance change that exceeds the given threshold. The DVS has shown its performance in a variety of reactive robot applications. [105] extracted visual features such as lines from the artifical environment to navigate a driverless car. Mueggler *et al*. proposed an evasive manoeuvre with quadrotors using the DVS [106]. Collision avoidance of fast-moving objects is achieved in this project. In terms of robot navigation, Maqueda *et al*. utilised deep learning for predicting the steering on the self-driving cars and showed a better steering performance than using standard

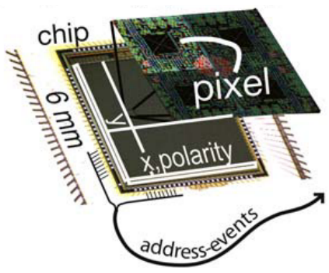


Figure 1.16: DVS chip micrograph. Picture from [4]

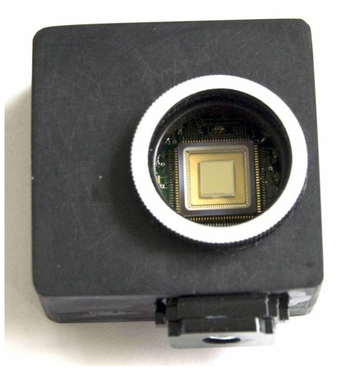


Figure 1.17: DVS128 sensor. Picture from [5]

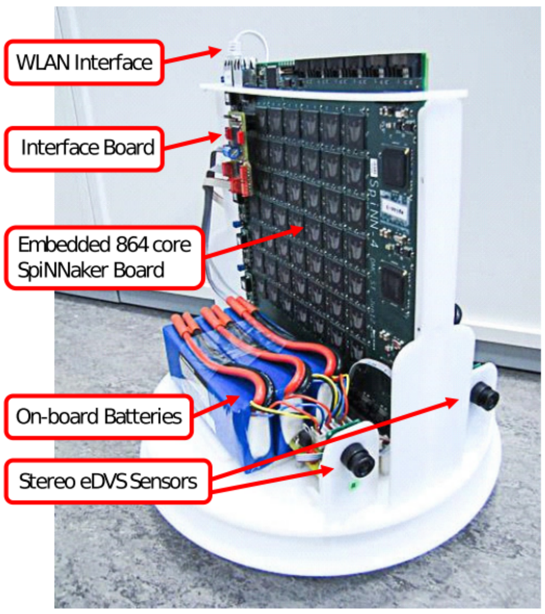


Figure 1.18: The SpiNNaker robotic System, consisting of two DVS cameras [6].

cameras [107]. Galluppi *et al*. developed an autonomous mobile robotic platform using two DVS sensors and a SpiNNaker computing system [6] (Figure 1.18), demonstrating a reactive behaviour with a speed less than 0.5 m/s.

However, the DVS-based vision system still needs to transfer image data to a separate computer to deal with image processing, such as reconstructing a whole image [108]. On the contrary, the Pixel Processor Array is able to perform all image processing on the image plane and there is no need to transfer images. Even though a DVS [105] is able to realise low-latency image processing and reduce redundant information transfer compared to a traditional frame-based camera, it is only sensitive to light change, thus it can not be used in static scenes.

Different from these reactive navigation mentioned above, this work, for the first time, uses the SCAMP PPA on the non-holonomic mobile platform to perform a reactive behaviour by carrying out associated actions after recognising the specific patterns (Figure 2.7) that achieves a faster navigation performance compared to these unconventional sensors.

### 1.6.2 Research Progress on Neural Networks with a SCAMP PPA

Research on neural network inference with the SCAMP PPA has been active in recent years. Table 1.3 lists the main research work in the area of neural networks, which covers fully convolutional neural networks and binary convolutional neural networks using DREG or AREG with various datasets and applications. High-level image processing, such as object classification, localisation and segmentation on sensor, is achieved with the neural network. The deployment of neural network onto the PPA is a breakthrough since it enables the PPA open to more possibilities with universal methods, which is unlike the conventional development methods with some combinations of low-level image processing methods for specific tasks. With the use of CNN, several types of tasks, such as classification, regression, localisation, and segmentation, can be feasible, hence enabling more applications. Table 1.3 shows the neural network-related work based on the SCAMP PPA vision system.

The research on CNN implementation and inference within PPA is pioneered by Bose *et al.* [110] where a CNN with a single convolutional layer performed upon the PPA array and a fully-connected layer upon its controller chip (M0). They performed 16-bit image convolution operations using $4\times4$ DREG "Super Pixel" blocks and demonstrated live digit classification based on MNIST dataset at around 200 FPS. In their work, the ternary {-1, 0, 1} kernel filters are stored on the flash (M4) of the PPA system, and are effectively encoded in the instructions/operation sent to the PPA array, performing convolutions sequentially. Furthermore, a mobile car localisation task is then explored using synthetic datasets, where the pre-processed edge information is mainly the clues for network inference. Notice that the localisation is realised by classifying the car's position along the $x$ and $y$ axis, respectively. To fully take advantage of PPA's parallel computing characteristics and further improve the CNN inference efficiency, Bose *et al.* [64], for

Table 1.3: Different convolutional neural network implementation with SCAMP and performance comparison.

| Network | Filter number | Layers (Conv + FC) | Dataset | Accuracy | frame rate | On-sensor/Near-sensor |
|---|---|---|---|---|---|---|
| Bose [63] | 16 5×5 | 1 + 1 | MNIST | ≈ 94.2% | 210 FPS | 1Conv/1FC |
| Bose [64] | 16/64 4×4 | 1 + 1 | MNIST | ≈ 93% | >3000 FPS | on sensor |
| – | 16+16 4×4 | 2 + 1 | MNIST | 92%~94% | 224 FPS | on sensor |
| Liu [65] | 64 4×4 | 1 + 1 | 8 Plankton | 80.5 % | 4016 FPS | on sensor |
| – | 16 4×4 | 1 + 2 | 8 Hand gestures | < 98.7% | 2092 FPS | 1Conv + 1FC / 1FC |
| – | 16 4×4 | 1 + 1 | Roshambo | < 97.73% | 8264 FPS | on sensor |
| – | 64 4×4 | 1 + 1 | 0,1 in MNIST | < 99.1% | 17543 FPS | on sensor |
| Liu FCN [66] | 16+64 4×4<br>64 1×1 | 3 + 0 | Simulation | - | 283 FPS | on sensor |
| Liu Binarized CNN [109] | 16+64 4×4 | 2 + 2 | EMNIST | < 86.74% | 178 FPS | 2Conv + 1FC / 1FC |
| Chen [67] | - | 0 + N | collected indoor | - | - | near sensor |
| AnalogNet [68] | 3 3×3 | 1 + 3 | MNIST | 96.9% | 2260 FPS | 1Conv/2FC |

the first time, proposed the idea of in-pixel weight storage, where the network's weights are directly stored within the registers of the PPA's PEs. This enabled both parallel computation of multiple convolutions, and implementation of a fully connected layer upon the PPA array, resulting in a ×22 faster CNN inference (4464 FPS) on the same digit recognition task. Based on these two works, Chapter 3 of this thesis further [65] proposes a high-speed lightweight neural network using BinaryConnect [89] with a new method for computing convolutions upon the PPA, allowing for varying convolutional strides. This work demonstrated four different classification tasks with frame rates ranging from 2,000 to 17,500 per second with different stride setups. Later, based on this network, a direct servo control using CNN results [111] and a simulated robot tracking from a drone [112] with on-sensor CNN computing results are exploited. In addtion, the AnalogNet2 [68, 113] extends the earlier work in [114], implementing a CNN which reaches 96.9% accuracy on the MNIST dataset at a speed of 2260 fps. However, their method requires all fully connected layers to be performed externally to the PPA array with only 3 convolutional kernel filters implemented in sequence on the PPA as the first layer. More kernel filters would significantly slow down the inference process. Notice that, in our work [67], a recurrent neural network is implemented on the micro-controller with features extracted on sensor. In this manner, the fully-connected layer of a neural network can be deployed similar with conventional embedded device. It is notable that Martel *et al*. trained a neural network of exposure time for each individual pixel off the sensor for HDR imaging and video compressive sensing [57].

Furthermore, this thesis (Chapter 4) [109] binarized CNN with batch norm both for classification and coarse segmentation. To deal with the classifications application with more labels and more segmentation tasks, we propose the idea of dynamic model swapping by uploading weights of trained models in sequence or according to the last inference result, targeting multiple sub-tasks decomposed from a more sophisticated task. We then demonstrate a servo control directly using the CNN inference results (Chapter 4) [111], which potentially indicates that motion control platforms, such as a ground vehicle or drones can have a light-weight servo control system without using external control units in the future.

Notice that the preceding neural network-related work mainly focuses on classification or classification-based localisation, both of which require fully connected layers. However, the parameters in fully-connected layers are typically substantially larger than those in convolutional layers due to the dense connections of each individual neuron. Thus, this thesis developed a fully-convolutional neural network (FCN) [66], not only presenting on-sensor image segmentation and localisation but also eliminating dense layers for a smaller memory footprint (Chapter 5).

## 1.7 Thesis Outline, Contributions and Publications

### 1.7.1 Thesis Outline

The outline of the thesis is as follows:

Chapter 1 provides the motivation for the study, the introduction to the study, and preliminary knowledge of on-sensor computer vision, neural networks, the PPA and SCAMP vision system to this thesis.

Chapter 2 illustrates the baseline parallel computer vision algorithms for the SCAMP PPA and applications based on the publications of [7, 45, 67], where the author made contributions. This chapter mainly covers three types of robot navigation tasks: reactive navigation, proximity estimation, and 1-D localisation based on the proposed image processing algorithms on the sensor. Additionally, we detail these parallel image processing algorithms on the SCAMP PPA and demonstrate their applications to mobile robots.

Chapter 3, 4, and 5 are the on-sensor neural network sections for high-level image inference. Chapter 3 illustrates the on-sensor neural network with binary weights based on the publication [65], where a new parallel and efficient method to implement the image convolution is proposed. With this new method, the convolution stride can be flexibly tuned for a range of tasks requiring a different level of speed. In addition, a direct servo control system with instructions from on-sensor neural network inference is demonstrated based on the proposed neural network with the publication of [111].

The binary neural network is then further explored, culminating in Chapter 4 exhibiting a fully-binarised CNN with binary weights and activations based on work [109]. A new neural network architecture is proposed to implement a more sophisticated neural network than previously possible. Based on the binarised activations, a new fully-connected layer is proposed by counting the number of bits, which is more accurate than earlier work.

Chapter 5 introduces a new fully convolutional neural network architecture based on the binarized neural network and then illustrates the implementation of different convolutional layers. Finally, applications such as coarse segmentation and localisation based on the paper [66] are demonstrated.

Chapter 6 summarises this thesis and presents the limitations, discussions and future directions.

Lastly, based on the proposed new binarized network, Chapter A develops a semi-simulated platform with the publication of [112] which establishes the communication between a real SCAMP vision system with the robot simulator in order to investigate and evaluate the robot-related applications. In addition, other useful tools and algorithms for SCAMP PPA are illustrated in Chapter A.

### 1.7.2 Contributions

The main and additional contributions of this work can be summarised as follows:

**Main contributions:**

1. **Machine Vision Algorithms for Agile Reactive Navigation:** We present a robot reactive navigation scheme with on-sensor computing to drive a ground vehicle run through a pre-set course of gates in a cluttered environment. No external hardware is needed to perform image processing to detect the target as conventional computer vision system does: perception and processing are performed on the sensor with low energy cost and high processing efficiency. The on-sensor image processing algorithms can run up to 200 fps indoors, enabling an average navigation speed of 2.2 m/s of gate passing through and 3.88 m/s for reactive obstacle avoidance.

2. **On-sensor Binary CNN for High-speed Classification:** We present a new image convolution implementation method for the PPA, incorporating variable convolution stride to allow for more efficient CNN inference, increasing the inference speed across various tasks depending upon the task's level of complexity. This study demonstrates that SCAMP-5 CNN can be implemented across a broader and more complex set of tasks, which had predominately focused upon only demonstrating MNIST classification. We demonstrate real-time hand gesture recognition, plankton classification from the National Data Science Bowl plankton dataset along with digit recognition. PPA inference speed for our approach is extremely fast across all tasks, ranging from 2000 to 17500 fps.

3. **Binarized CNN with Batchnorm:** We propose, train, and demonstrate our experiment using a completely binarized network (both binary weights and activations)

specifically for PPAs. This approach of binary activations addresses the accumulation of analogue computing errors and value saturation after each layer, thus enabling deeper networks while preserving performance.

4. **Binarized FCN for Localisation and Segmentation:** We present the first implementation of an FCN architecture for PPAs. Our approach uses group convolutional layers and stores hundreds of convolutional filter weights upon the focal plane of the PPA. Unlike earlier work, we apply batch normalisation during training and utilise this to learn bias parameters applied during inference on the PPA device. We provide the first demonstration of object localisation and coarse segmentation tasks on a PPA, with previous works being only concerned with classification tasks.

5. **CNN Tree Architecture with Multiple Networks for Many Labels on-Sensor:** With the binary neuron activations as inputs, the linear layers can be implemented by simply counting the bit number rather than using an approximate method in earlier work [65]. We propose a CNN tree architecture where multiple neural networks can be composed for more sophisticated inference tasks by dynamically uploading neural network models onto the sensor. We explore more complicated tasks on the PPA across 37 English letter classification and provide the first demonstration of object localisation and coarse segmentation tasks on a PPA, with previous works only concerned with classification tasks.

**Additional contributions:**

1. **Semi-simulated Platform and Direct Servo Control with PPA:** We developed a simulated environment and datasets (available from github[3]) to support PPA developers and researchers for idea validation off and on-sensor. In addition, we demonstrate direct visual sensory-motor control using high-speed CNN inference via a SCAMP-5 Pixel Processor Array (PPA). A binary Convolutional Neural Network (CNN) is used for classic rock, paper, scissors classification problems at over 8000 FPS. Control instructions are directly sent to a servo motor from the PPA according to the CNN's classification result without any other intermediate hardware.

2. **Collaboration work: Proximity Estimation [115] and on-sensor Mapping & Localisation [7]:** We propose to combine the on-sensor edge, blob, corner points,

[3]https://github.com/yananliusdu/scamp5d_interface

motion parallax extraction and send extracted features into a recurrent neural network for proximity estimation with an obstacle avoidance application on a mobile ground vehicle. In addition, we present the 1-D on-sensor mapping and localisation with a PPA. By comparing input images and database on the sensor and the PPA motion model, the on-sensor algorithms run up to 300 Hz on large public datasets.

### 1.7.3 List of Publications, Submissions, and Press

The work illustrated within this thesis has been presented in the following peer-reviewed publications[4], under-reviewing submissions, pre-print, and manuscripts. In addition, some of our research outputs can also be seen from the press.

**Publications**

1. **Yanan Liu**, Laurie Bose, Colin Greatwood, Jianing Chen, Rui Fan, Piotr Dudek, Thomas Richardson, Steven J. Carey, and Walterio Mayol-Cuevas, "Agile reactive navigation for a non-holonomic mobile robot using a pixel processor array", *IET Image Processing*, 2021. (Chapter 2)

2. **Yanan Liu**, Jianing Chen, Laurie Bose, Piotr Dudek, and Walterio Mayol-Cuevas, "Direct servo control from in-sensorcnn inference with a pixel processor array," in *2021 IEEE International Conference on Robotics and Automation Workshop (ICRA-W): On and Near-sensor Vision Processing, from Photons to Applications,* ***Oral presentation,*** IEEE, 2021. (Chapter 3)

3. **Yanan Liu**, Jianing Chen, Laurie Bose, Piotr Dudek, and Walterio Mayol-Cuevas, "Bringing a robot simulator to the Scamp vision system," in *2021 IEEE International Conference on Robotics and Automation Workshop (ICRA-W): On and Near-sensor Vision Processing, from Photons to Applications,* ***Best poster/video,*** IEEE, 2021. (Chapter Appendix)

4. Hector Castillo-Elizadle, **Yanan Liu**, Laurie Bose, and Walterio Mayol-Cuevas, "Weighted node mapping and localisation on a pixel processor array," in *2021 IEEE International Conference on Robotics and Automation (ICRA),* IEEE, 2021. (Chapter 2, in this paper, the author acts as a co-supervisor of Hector's MSC project and contributes to LBP-based image feature extraction & localisation methods, and parts of experiment.)

[4] https://scholar.google.com.hk/citations?user=7otPL_QAAAAJ&hl=en

5. **Yanan Liu**, Laurie Bose, Jianing Chen, Steven J. Carey, Piotr Dudek, and Walterio Mayol-Cuevas, “High-speed light-weightcnn inference via strided convolutions on a pixel processor array,” *The 31st British Machine Vision Conference (BMVC) : 7th - 10th September 2020* (Chapter 3)

6. Jianing Chen, **Yanan Liu**, Steven J. Carey, and Piotr Dudek, “Proximity estimation using vision features computed on sensor,” in *2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 2689–2695,* IEEE, 2020. (Chapter 2, in this paper, the author contributes to two types of feature extractions and parts of the free-roaming experiments.)

**Submissions and Manuscripts**

1. Piotr Dudek, Thomas Richardson, Laurie Bose, Steven J. Carey, Jianing Chen, Colin Greatwood, **Yanan Liu**, and Walterio Mayol-Cuevas, “Pixel processor array for sensor-level computer vision on agile robots,” *Science Robotics*, under review. (In this paper, the author contributes to the robot agile reactive navigation, high-speed neural network inference, and SCAMP PPA-based sensory-motor systems.)

2. **Yanan Liu**, Laurie Bose, Jianing Chen, Rui Fan, Pitor Dudek, and Walterio Mayol-Cuevas, “On-Sensor Binarized CNN Inference with Dynamic Model Swapping in Pixel Processor Arrays,” *Frontiers on Neuroscience*, 2022, under review. (Chapter 4)

3. **Yanan Liu**, and Walterio Mayol-Cuevas, “On-sensor machine vision with a pixel processor array: a review,” 2022, under submission. (Chapter 1)

**Pre-print**

1. Wen Fan, **Yanan Liu**, Yifan Xing, “Fully-simulated Integration of Scamp5d Vision System and Robot Simulator,” 2021, *arXiv preprint arXiv:2110.06386*. (Chapter Appendix, in this work, the author contributes to the architecture of the fully-simulated system framework and Robot simulator interface to SCAMP simulator.)

2. **Yanan Liu**, Laurie Bose, Lu Yao, Pitor Dudek, and Walterio Mayol-Cuevas, “On-sensor binarized fully convolutional neural network with a pixel processor array,” *arXiv preprint arXiv:2202.00836* (Chapter 5)

**In the Press**

1. University of Bristol, University of Manchester, News and Features: **Cameras that can learn**, 13 October 2020[5,6]

2. University of Bristol, News and Features: **Bristol researchers create a camera that knows exactly where it is**, 1 June 2021[7]

3. Vision spectra: **Camera-processor Chip Brings Computer Vision Closer to Natural Perception**, January 2021[8]

**Experimental Videos**

All the experimental videos demonstrated in this thesis can be seen from https://www.youtube.com/channel/UCEawLcWSXtrxeg8jDamRApw/videos

[5]https://www.bristol.ac.uk/news/2020/october/scamp.html

[6]https://www.manchester.ac.uk/discover/news/cameras-that-can-learn/

[7]https://www.bristol.ac.uk/news/2021/june/camera-maps-where-it--s.html

[8]https://www.photonics.com/Articles/SCAMP_Brings_Computer_Vision_Closer_to_Natural/p22/vo209/i1317/a66564

CHAPTER 2

# On-Sensor Baseline Parallel Machine Vision Algorithms

Conventional machine vision systems usually consist of independent hardware in charge of sensing, storage, transmission, processing, and execution, respectively, to meet the needs of interchangeability. The visual sensors in a conventional machine vision system are usually used to 'duplicate' the world as much as possible and deliver these raw data to processors for later processing, leading to increased latency, power consumption, and system complexity. With these issues in mind, this chapter explores a new machine vision system and its associated parallel algorithms by introducing SCAMP PPA, which features low power consumption and latency by incorporating sensing, storage, and computing. The SCAMP vision system has been developed and studied in domains including image processing, computer vision, robotics, neural networks for around 20 years. This is because, PPA, compared to other existing conventional computer vision systems, facilitates vision systems with on-sensor image processing hence reducing system complexity, lowering power consumption, and boosting information processing efficiency. There is a rich history of research on machine vision systems in hardware platforms and their associated algorithms as introduced in the last chapter.

## 2.1 Introduction

This chapter demonstrates the adaptability of the machine vision algorithms created in this thesis for a variety of targets by exhibiting applications throughout robot agile, reactive navigation, obstacle avoidance, and one-dimensional mapping and localisation. First of all, the high-speed pattern recognition on the sensor is developed and then integrated with a non-holonomic RC mobile robot system. With this hardware platform, the RC car can be instructed by on-sensor processed results to run through 'gates' at high speed. Then, to further explore the image processing on more general tasks, we extract common image features, such as edges, corner points and blobs, to a recurrent neural network to estimate the proximity from the environment, which enables an RC car to navigate indoors without collisions to the ambient objects. Finally, a one-dimensional mapping and localisation system on the sensor is presented to deal with the general robot localisation problem. This chapter shows the various parallel image processing algorithms across different applications. Both the proposed on-sensor image parallel processing algorithms and applications prove the effectiveness of the SCAMP PPA on machine vision tasks.

## 2.2 Machine Vision for Agile Robot Reactive Navigation

This section is mainly based on the publication of [45], presenting an agile reactive navigation strategy to drive a non-holonomic ground vehicle around a pre-set course of 'gates' in a cluttered environment using a low-energy-cost processor array sensor. This enables machine vision tasks to be performed directly upon the sensor's image plane, rather than using an external general-purpose computing unit. This section demonstrates a small ground vehicle running through, avoiding multiple 'gates', or tracking a predefined pattern at high speed using minimal computational resources. To achieve this, parallel image processing algorithms are developed for the PPA and captured images are then processed directly on the vision sensor acquiring target information for controlling the ground vehicle. The image processing algorithm can run at up to 200 fps at indoor illumination levels. Conducting image processing at the sensor level avoids the bottleneck of image transfer encountered in conventional image sensors. The real-time performance of on-board image processing and robustness is validated through experiments. Experimental results demonstrate that the algorithm's ability to enable a ground vehicle to navigate at an average speed of 2.20 m/s for passing through multiple gates and 3.88 m/s for a ‘slalom’ task in an environment featuring significant visual clutter.

### 2.2.1 Introduction

Vision-based mobile robot navigation technology plays a vital role in the field of intelligent transportation systems and robotics, being increasingly used across many fields of industry, such as, driverless cars [116], assisted living, logistics [117], and domestic applications [118]. However, implementing a mobile robot to perform navigation tasks rapidly, robustly and energy-efficiently can be challenging especially in cluttered environments [119]. A reactive navigation methodology can possibly be a solution to this problem that combines sensory information with robot control directly [120] through the processing of sensor data [121].

A number of approaches for visual reactive ground navigation have been proposed in recent and past years. In [122], Penin *et al.* utilised a Unmanned Aerial Vehicle (UAV) to perform visual-based reactive navigation, track targets and keep a fixed distance from the target at a speed of 1.00 m/s. Galluppi *et al.* [123] developed an autonomous mobile robotic platform using two Dynamic Vision Sensors (DVS) and a SpiNNaker computing system. According to their work, the average navigation speed is less than 0.5 m/s. Many of the existing platforms require expensive computation to perform reactive navigation, which limits reaction time to changing environments and the mobile robot’s top speed [124–126]. More recent vehicles geared for learning visual navigation, e.g., DeepRacer[1] or Audi driving cup cars[2], are slow in processing and are yet to demonstrate agile behaviours. Dedicated sensors for car line following do allow faster operation but are tuned for the task and thus less generic.

In agile visual reactive navigation, a vision sensor with a high frame rate is necessary to enable the mobile robot to react to the environment quickly. Various types of visual sensors, such as CCD/CMOS cameras [13] and RGB-D sensors [127], are widely used for map building, robot localisation, navigation and other vision guidance applications [13, 128–130]. However, these sensors transmit entire images to a computer for processing, a process that requires relatively significant time and power, decreasing the response time of the robot [43].

The key contributions of this section compared to the previous work [131] includes: a) improving iterative flooding operations to enhance its robustness when extracting

[1]https://aws.amazon.com/deepracer/
[2]https://www.audi-autonomous-driving-cup.com/

disks;b) designing a filtering algorithm based on the neighbourhood pixel values to denoise the image; c) exploiting the PID control to complete the navigation task and integrating the improvements mentioned above to directly link the visual information from the image plane to robot control instructions enabling agile reactive navigation. Compared to a traditional visual sensor, each pixel in the Pixel Processor has storage and processing abilities, which supports fast and lower-power parallel computation directly on the visual sensor. The PPA is capable of performing general-purpose vision algorithms at several thousand frames per second due to its parallel computing features [10], [108]. This section presents a novel agile reactive navigation strategy using the PPA and a mobile robot with a low-priced single-board computer and a simple control structure. In our applications, a car-like mobile robot self-navigates, running through multiple gates at an average speed of 2.20 m/s according to the guidance provided by visual information extracted from the pre-designed patterns pasted on the gates (see Figure 2.7(a)). The control instructions for the mobile robot are generated by comparing the desired image and the current image. The mobile robot also achieves a highest speed of 3.88 m/s during a 'slalom' process (Figure 2.7(b)). In particular, the mobile robot is directly instructed by the coordinates of feature points on the patterns themselves, hence there is no need to perform transformation calculations among camera coordinate system, image coordinate system and ground vehicle coordinate system.

The remaining sections of this work are organised as follows: subsection 2.2.2 describes the proposed vision system including the introduction of the SCAMP-5 vision system and image processing algorithms. In subsection 2.2.3, we present the robot control system which consists of hardware and software structure. Subsection 2.2.4 illustrates the agile reactive navigation method. Experimental results are illustrated and the proposed system performance is evaluated in subsection 2.2.5. Finally, subsection 2.5 summarises this work.

### 2.2.2 Gate Pattern Recognition Algorithm

This subsection describes the SCAMP-5 algorithm used to detect the pre-designed patterns. Patterns are a common means to give instructions to human drivers or even to autonomous cars, such as in line following. The challenges include tolerance to noise, clutter and fast enough processing. The patterns we use are primarily aimed at demonstrating agile visuo-control behaviours.

For the gate's task, each pattern contains four black disks surrounded by two black concentric squares. The disk coordinates in the image plane are utilised to adjust the

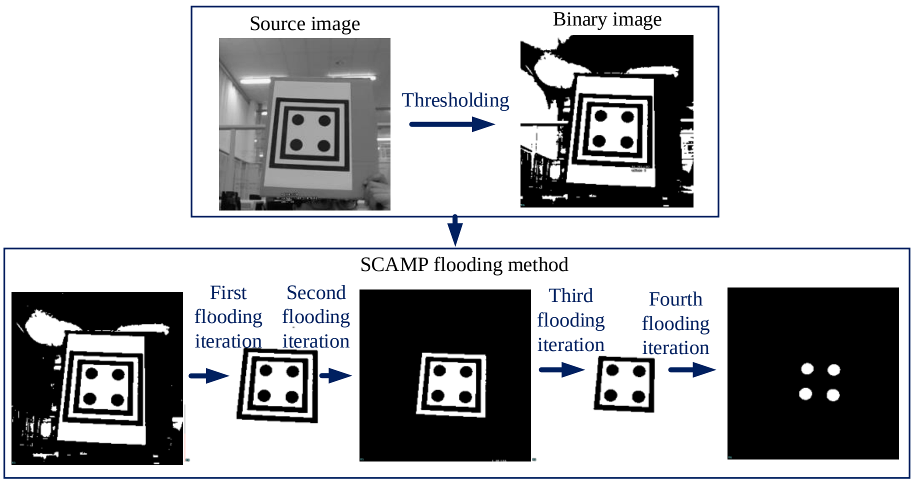


Figure 2.1: Pattern extraction procedure. The four extracted dots in the final result form a quadrangle.

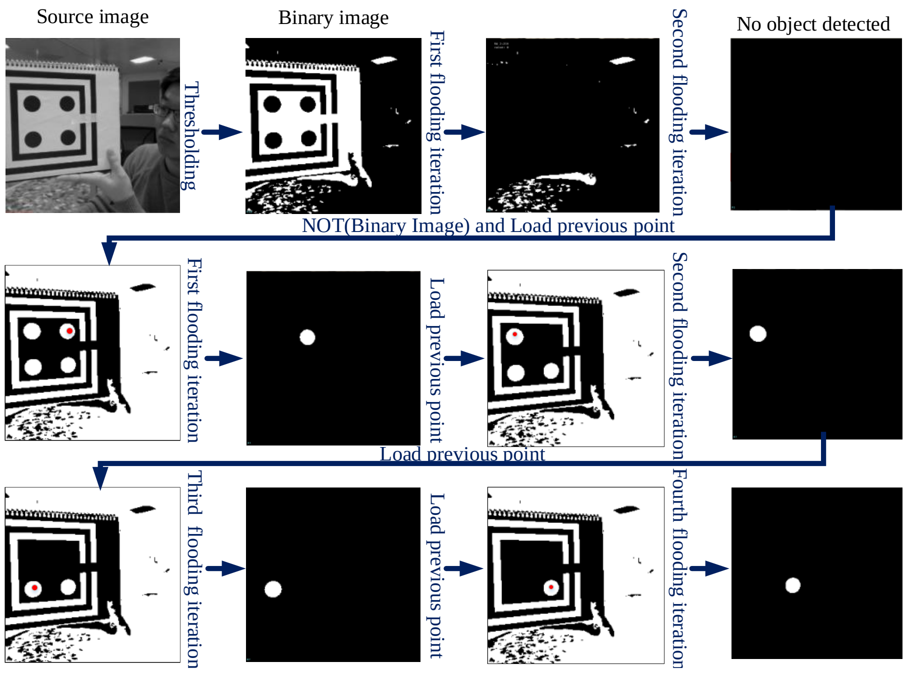


Figure 2.2: Image processing procedure in which the squares are broken.

rover's position and orientation, guiding it to go through multiple gates at high speed. Figure 2.1 and Figure 2.2 show the image processing procedure conducted on SCAMP-5. High-frame-rate real-time image processing is typically challenging to conduct with a standard camera and computer setup due to the limited rate of image capture and delay in data transfer that such devices have. In this work the target extraction algorithm is designed to exploit SCAMP-5's fast image flooding ability to minimise computation time per image and improve the image processing robustness. Iterative flooding and bit NOT operations are used to extract centres of four disks out of the cluttered background [131]. This method fills regions with value '1's except these areas closed by value '0's. Hence those areas that are out of the closed regions would be filled with '1's and with this method, the background of the pattern is easily eliminated within several iterations. As shown in Figure 2.1, after the first flooding iteration, the background is eliminated. Therefore, the time cost for this kind of operation is much less than that of the traditional camera, because this flooding operation is carried out in all Processing Elements (PEs) in parallel and asynchronously rather than pixel by pixel for traditional image processing pipeline.

The extraction of four points in the pattern relies on the presence of two black concentric boundaries. However, issues, such as target occlusion, bright light reflections or simply the target being distant from the camera, may break or merge together those two boundaries, causing failure of the dot extraction process. For example, in Figure 2.2, the pattern in the source image is partly outside the view field and the boundaries broken on the right hand side for demonstration. In this case, the direct flooding method is not useful since four disks are not enclosed by '0's. To improve the robustness of the pattern extraction, the prior knowledge of where the dots were located in the previous frame is used whenever the two black boundaries are not present. This is illustrated in Figure 2.2, when there is no object remaining after two flooding and inversion operations. Firstly, the inversion of the current binary image is performed. Then, a point from the last detected disk centre is loaded into the current frame. Since the frame rate of the SCAMP-5 is set to 200 fps or more, the shift between two consecutive frames is small, and thus, we assume that the loaded point falls into the current corresponding disk. Then, flooding is conducted outwards from this point using the current inverted image as a mask. If the point from which flooding was performed existed within the disk, the resulting image contains only that disk which can then be easily extracted. After getting rid of the first extracted disk from the inverted image, the location of the remainder of the disks can be obtained using the loading point and flooding method iteratively. The detailed algorithm description can be seen in Algorithm 1.

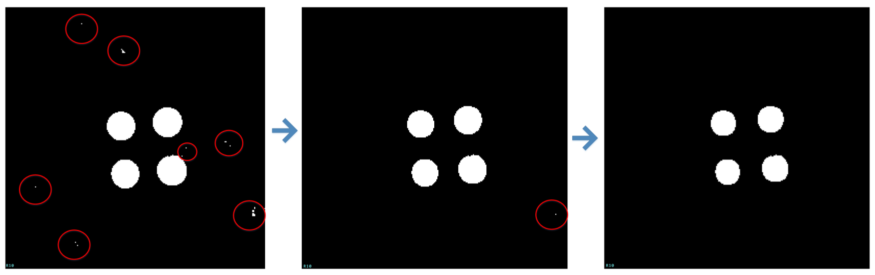


Figure 2.3: Image de-noising, the shrink radius is 2 pixels. Noisy pixels are circled in red. All noisy pixels are eliminated in the third image.

**Algorithm 1:** Disk extraction based on the SCAMP-5 vision system

```
1  INPUT:
2  Threshold // used to get binary image
3  Iterations // iteration times for flooding and inversion operation
4  Disc_num // pre-set number of disks
   Result: Coordinates // coordinates of four disks' centres
5  while true do
6      R5 = Scamp5_get_image()
7      R6 = Scamp5_threshold(R5,Threshold)
8      for n = 0 to Iterations do
9          R7 = Scamp5_flood(R6)
10         R6 = AND(R6, NOT(R7))
11         R6 = NOT(R6)
12     end
13     Image_filtering (see Algorithm 2)
14     Num = Object_detect(R6)
15     if Num == Disc_num then
16         Coordinates = Scamp5_scan_boundingbox(R6)
17     else
18         R7 = Scamp5_load_point(previous_point)
19         R7 = Scamp5_flood(R6)
20         R8 = R6 XOR R7
21         R6 = R8
22         Coordinates = Scamp5_scan_boundingbox(R7)
23     end
24     previous_point = Coordinates returnCoordinates
25 end
```

Noise in an image is caused by various factors, such as overexposure, cluttered environment and quantisation. In this scenario, the light spots caused by overexposure or reflection are the main sources of noise. To eliminate the noise, an image filtering method of erosion and dilation based on the neighbour communication is conducted upon SCAMP-5. Since each processing element can communicate with its four neighbours (north, south, west, and east), a given pixel can access its neighbours' data directly. Four new pictures are obtained after moving the source image into four directions with a pixel distance $P_{\text{step}}$. Then using 'AND' operator to add these four images together on the image

**Algorithm 2:** Image filtering with erosion and dilation

```
1  INPUT: $P_{step}$
   Result: R10
2  Erosion procedure
3  R5 = Scamp5_get_image()
4  R6 = Move R5 to the north with $P_{step}$
5  R7 = Move R5 to the south with $P_{step}$
6  R8 = Move R5 to the west with $P_{step}$
7  R9 = Move R5 to the east with $P_{step}$
8  $R10 = R6 \cap R7 \cap R8 \cap R9$
9  Dilation procedure
10 $R5 \leftarrow R10$
11 R6 = Move R5 to the north with $P_{step}$
12 R7 = Move R5 to the south with $P_{step}$
13 R8 = Move R5 to the west with $P_{step}$
14 R9 = Move R5 to the east with $P_{step}$
15 $R10 = R6 \cup R7 \cup R8 \cup R9$
16 return R10
```

plane directly. In Figure 2.3, noisy area whose radius is smaller than shrink radius will be eliminated. The pseudo codes are given in Algorithm 2.

We now discuss the way of getting each dot's centroid in the image plane. Firstly, an inbuilt function *scan_event* of the SCAMP-5 is used to find the location of a white pixel in the image. This white pixel must be located within one of the disks we wish to extract. We then perform a flooding operation originating from this extracted point, using current image as a mask, resulting in an image consisting of the entire disk the point was within, as shown in the second image of Figure 2.4. The bounding box of all white pixels in the image (which in this case are those of the disk) is then extracted using another built-in SCAMP-5 function *scan_boundingbox*. After that, the disk is removed from the original binary image using a NOT operation. With this method, one white disk can be extracted. This process is similar to that used in Figure 2.2 where squares are not intact. The *scan_boundingbox* function outputs the centre position of the white region. After performing this type of operation for four iterations, all centre positions of the disks can be extracted separately.

The image processing time of all algorithms mentioned above is recorded by the SCAMP-5 application that receives the image processing information from the SCAMP-5 vision system through a USB cable. In this scenario, the frame rate is set to 200 fps given the illumination conditions in the indoor arena. It is noteworthy that the proposed

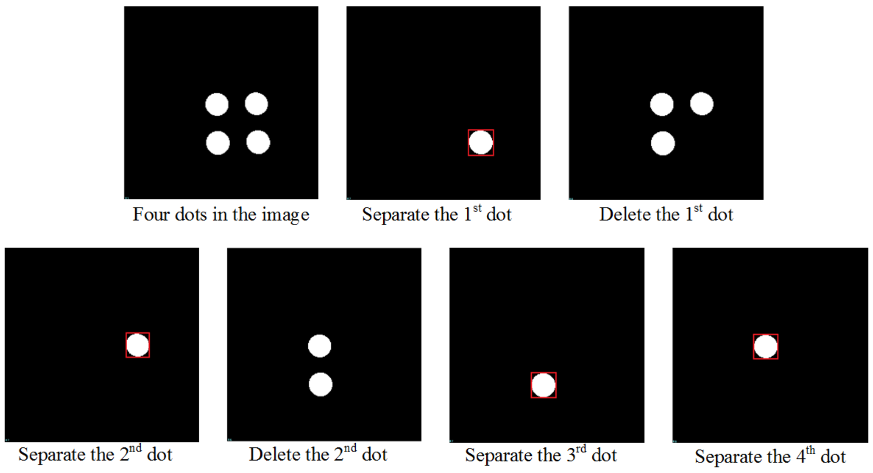


Figure 2.4: An example of getting coordinate of each dot's centre.

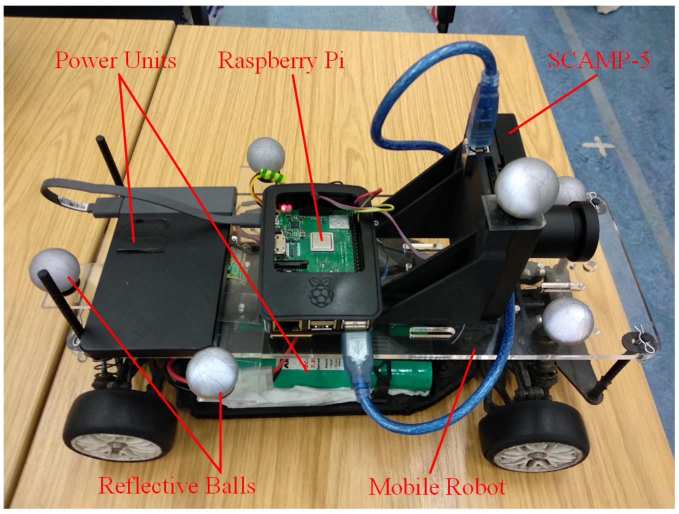


Figure 2.5: Subaru RC car (1:10 scale model) and hardware used in experiments.

image processing algorithm can be easily performed at more than 2000 fps with enough light illumination.

### 2.2.3 Architecture of Robot System

A model Subaru rover chassis, shown in Figure 2.5, was utilised for the agile navigation experiment. The rover was controlled by both the visual data from the SCAMP-5 and the remote control shown in Figure 2.6. Specifically, the visual data controls the steering while the speed of the car is set to a fixed value using the remote control. As shown in Figure 2.6, images are captured and processed on the vision chip and the processing

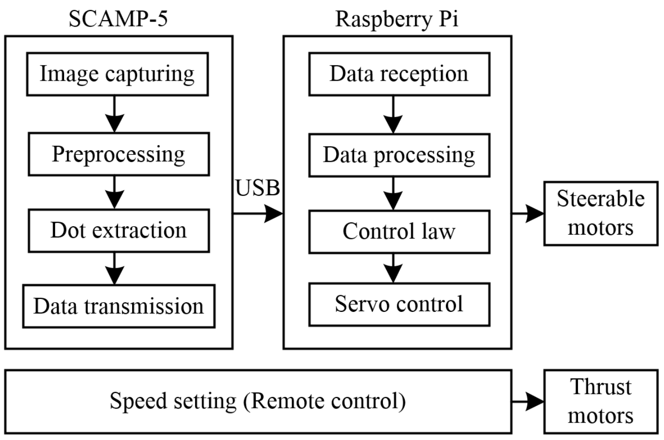


Figure 2.6: Control architecture of the navigation system.

| Hardware name | Hardware configuration |
|---|---|
| Subaru RC Car | SKU H94123-12344 |
| SCAMP-5 Vision System | SCAMP-5d (gray-level image, 256×256 array) |
| Raspberry Pi | Version 3 Model B+, CPU: 1.4GHz |
| Subaru power unit | ANSMANN, RC-Racing Pack |
| Raspberry Pi power unit | HUAWEI Colorphon 5 |
| Remote control | TARANIS |

Table 2.1: List of hardware adopted for the robot navigation system.

results are sent back to the chip controller. Finally, these coordinates are transferred to the Raspberry Pi, which handles read-out data and performs mobile robot navigation tasks. The detailed hardware and its hardware configurations can be seen in Table 2.1.

### 2.2.4 Vision-based Agile Reactive Navigation

Reactive navigation is one of the popular control paradigms in the area of mobile robotics, which features decision making through light processing of sensory data [132]. Different from deliberative paradigm where heavy image processing computation is needed to enable the mobile robot a higher level of intelligence in the complex environment, reactive navigation is suitable for a low-cost mobile robot platform with limited computational power and memory [132]. Mobile robot reactive navigation is useful in terms of completing tasks which do not require a map or localisation within a map. Furthermore, reactive navigation enables the mobile robot to perform navigation tasks with low-powered hardware. This type of navigation architecture generates control instructions only using the current sensory readings of the environment. It also builds a direct link between the

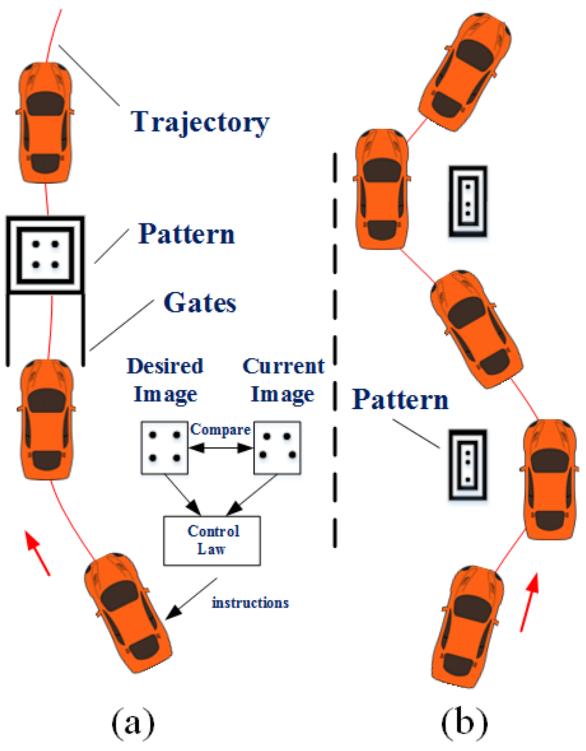


Figure 2.7: Examples of the proposed ground vehicle agile navigation. (a) Passing through the gates. The width of the car body and the gate is 18 cm and 36 cm, respectively; (b) the 'slalom' navigation.

actuator control and sensor readings. Consequently, there is no need to perform matrix transformation among camera coordinate, robot coordinate and world coordinate to get their relative pose. And building a complete map of the environment is not essential [133].

#### 2.2.4.1 Object Tracking

The SCAMP-5 PPA extracts the four disks located within any visible targets which are then utilised to steer the mobile robot through the gate with the observed target. Controls are generated by comparing the four disks extracted from a pattern in view against four reference disk for a pattern at a known distance and angle. However, when carrying out the disk extraction, the order of these four dots is unknown because the relative position between the camera and the gate can be random. Hence, a method is developed to make these eight points correspond to each other before making a comparison.

As shown in Figure 2.8, the centre of these four dots $P_{cc}$ can be obtained by $P_{cc}$ =

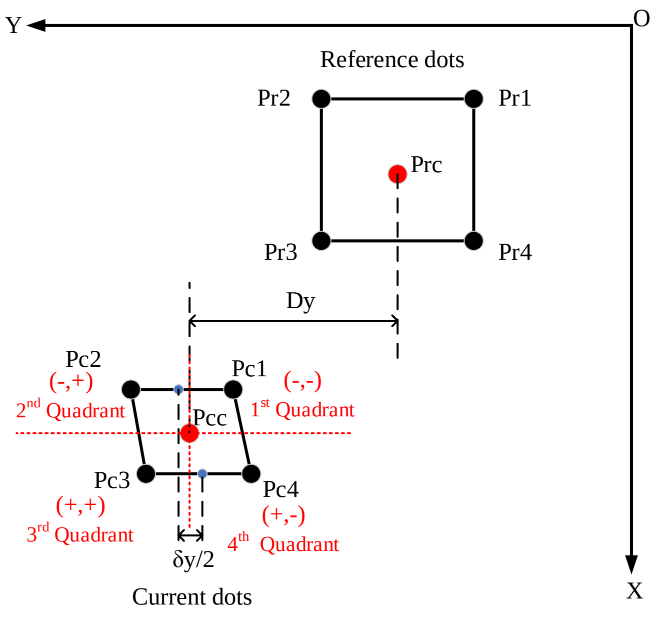


Figure 2.8: Quadrant allocation for four current dots and its difference from the reference dots. These dots represent the centres of extracted disks and they form a rectangle in reference image and a quadrangle in current image respectively.

$\sum P_{ci}/4$. By comparing the difference between $P_{cc}$ and $P_{ci}$, these four dots can be allocated into the corresponding quadrant. The current picture is divided into four quadrants each of which will contain a specific disk which can then be compared to the corresponding disk in a reference pattern.

$$Sign = P_{ciy} - P_{ccy}, \quad i \in \{1,2,3,4\}. \tag{2.1}$$

Where $Sign$ represents the sign symbol of the difference between $P_{ccy}$ and $P_{ciy}$ along $y$ axis.

#### 2.2.4.2 Mobile Robot Reactive Navigation

This subsection compares the difference in both position and shape between the reference rectangle and the currently processed quadrangle to generate instructions for the mobile robot to run through gates vertically. The reference image is defined with an image that was pre-set by putting the ground vehicle in front of the gate at a distance which is the minimum distance that is in sight of the sensor. The reference rectangle is compared to the currently processed quadrangle; from this comparison, controls are generated for the RC rover which will bring the observed rectangle closer to the reference, guiding it towards the gate. As the pose of the rover gets closer to that at which the reference image was taken, the similarity between the quadrangles correspondingly increases. This control guides the rover through the gate without collision enabling the rover to see the next gate.

During reactive navigation, the distance $D_{\mathrm{y}}$ between $P_{\mathrm{cc}}$ and $P_{\mathrm{rc}}$, the deformation $\delta y$ along y axis (see Figure 2.8) act as the input for the PID control.

$$D_{\mathrm{y}} = P_{\mathrm{cc}} - P_{\mathrm{rc}}, \tag{2.2}$$

where $D_{\mathrm{y}}$ is utilised to adjust the angle of the front wheels towards the target. $\delta y$ describes the deformation of the pattern along y axis caused by the relative position between the pattern and the rover. In this scenario, $\delta y$ is the difference between top side centre and bottom side centre in the quadrangle.

$$\delta y = P_{\mathrm{c3y}} + P_{\mathrm{c4y}} - (P_{\mathrm{c1y}} + P_{\mathrm{c2y}}), \tag{2.3}$$

where $\delta y$ is used as a compensation to slightly change the wheel angle and attempt to take the rover straight through the gate rather than going through at an angle. The adopted control method is as follows:

$$Output = Output_1 + Output_2, \tag{2.4}$$

$$Output_i = K_{Pi} e(k)_i + K_{Ii} \sum_{j=0}^{k} e(j)_i + K_{Di}[e(k)_i - e(k-1)_i] \tag{2.5}$$

$$e(k)_1 = D_{\mathrm{y}}, \tag{2.6}$$

$$e(k)_2 = \delta y. \tag{2.7}$$

Where, $i \in \{1,2\}$, $K_P$, $K_I$ and $K_D$ are coefficients adjusted experimentally in this paper using Ziegler-Nichols [134]. $e(k)_i$ is the error between $D_{\mathrm{y}}$, $\delta y$ and 0. As we can see from the equation, the visual information is slightly processed before generating control instructions for the mobile robot. The aim of this PID control is to minimise both $D_{\mathrm{y}}$ and $\delta y$ to 0. During this process, the ground vehicle is moving towards the gate while adjusting its pose to enter the gate head on.

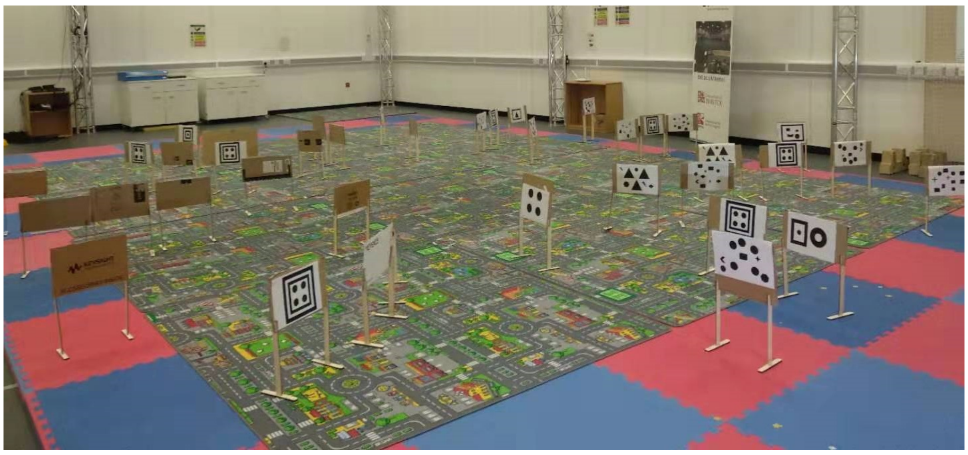


Figure 2.9: Gates and clutter layout in the arena.

### 2.2.5 Experimental Results

Experiments are carried out to test how agile the ground vehicle would track the target (the gate) and run through the gate in heavily cluttered environment. The mobile robot is set at a constant speed using the remote control. In Figure 2.9, there are eight gates placed in the robot arena taking into consideration space limitations and the vehicle's turning radius. Beside each gate, there are some other gates with a random or similar pattern to the real pattern acting as a disturbance to show the robustness of this navigation system to the cluttered environment.

The views in front of the rover are recorded by a GoPro camera. As we can see from Figure 2.10, the images captured by the GoPro are blurred because of the high speed and vibration of the chassis during navigation. However, the SCAMP is able to get clear images and output effective visual data. When the mobile robot is running through these gates, its trajectory is recorded by the VICON motion tracking system[3]. Figure 2.11 shows the rover's paths and the layout of eight gates and the disturbance. As we can see, the rover runs though these gates, which indicates the effectiveness of the proposed control method and the robustness of the image processing. Figure 2.12 shows the changes of the front wheel angle when the rover is approaching a gate and the steering angle is asymptotically close to 0. The extracted dots are recorded during the navigation and plotted. As shown in Figure 2.13, the relative position of these four dots is changing when the rover is running towards a gate. The difference between the reference image and the current image is used for controlling the motion of the mobile

[3]https://www.vicon.com/hardware/cameras/

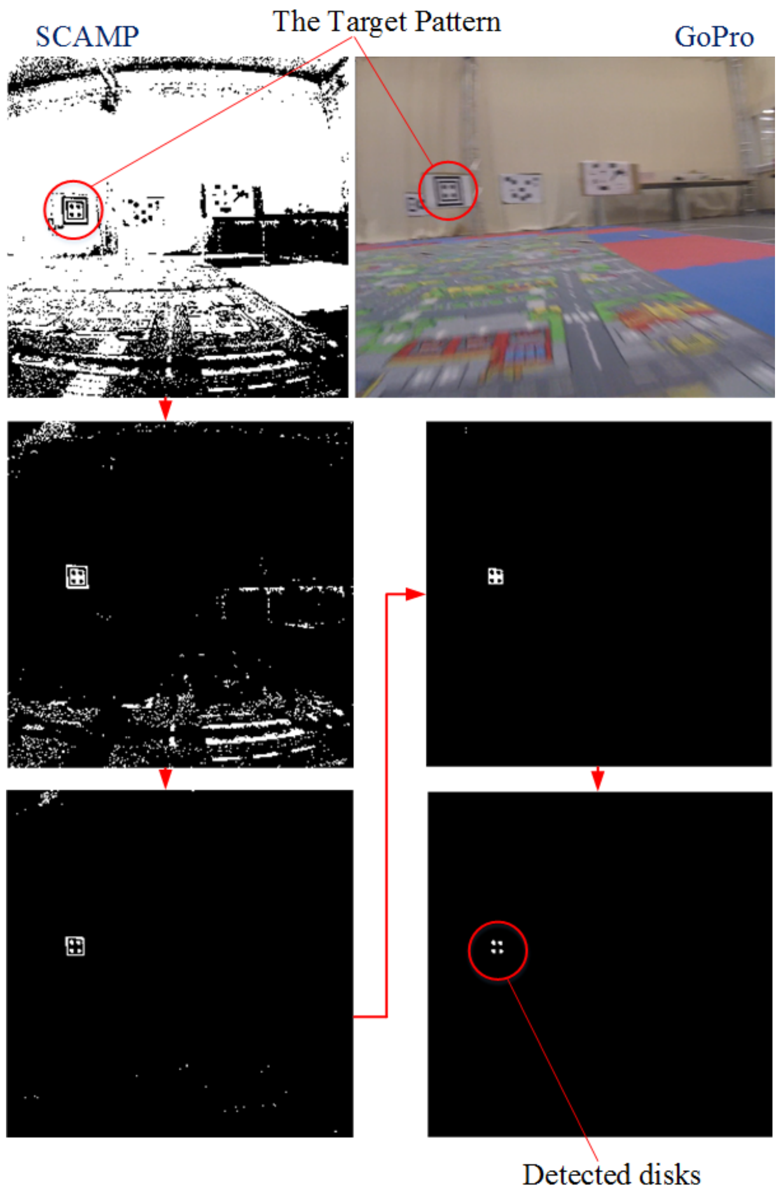


Figure 2.10: Image captured by the SCAMP-5 and GoPro during high-speed navigation. The SCAMP-5 detects four disks during high-speed navigation in a cluttered environment. Experimental video can be seen: `https://youtu.be/e85q-yoBuSk`

robot. Consequently, in the image plane, their difference should be getting increasingly smaller during the navigation process. The expected phenomenon can be seen in Figure 2.13, where the current pattern shape is becoming increasingly closer to the reference pattern, in terms of both position and shape.

Although all the patterns are designed identically, their positions on the gate are

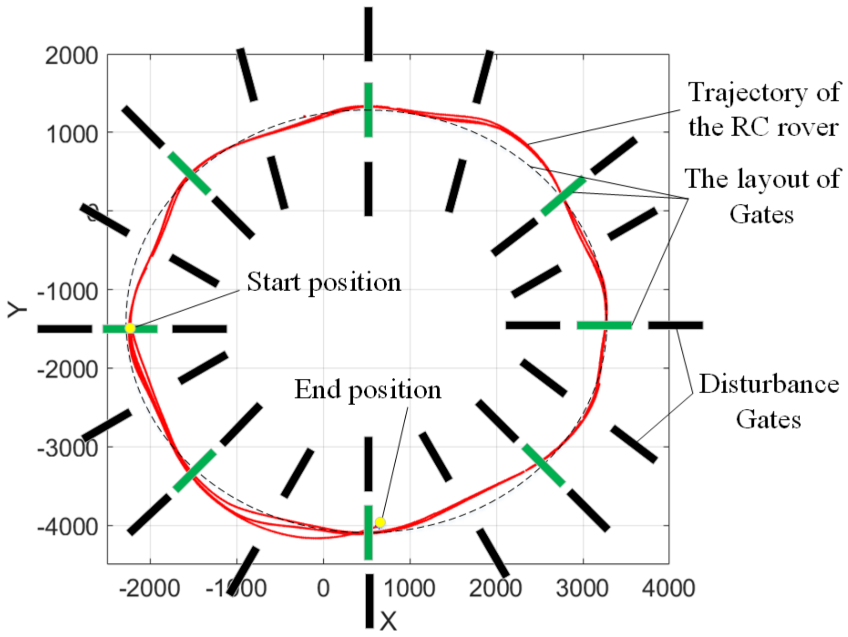


Figure 2.11: Trajectory in the arena.

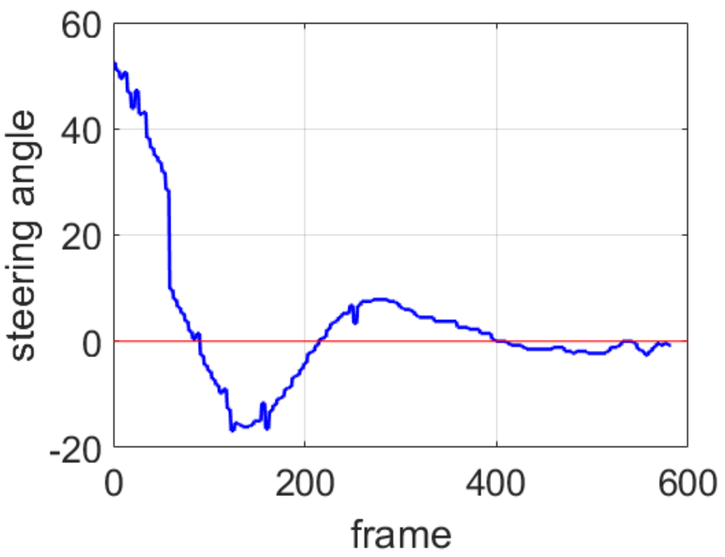


Figure 2.12: The angle change when the rover is approaching a gate.

slightly different. In addition, the view of the SCAMP-5 is possibly not horizontal because of its suspension system during high-speed motion. As a result, the final image when the mobile robot is about to pass through the gate could be slightly different from the reference image.

The agility of this reactive navigation can be seen from Figure 2.14. The average nav-

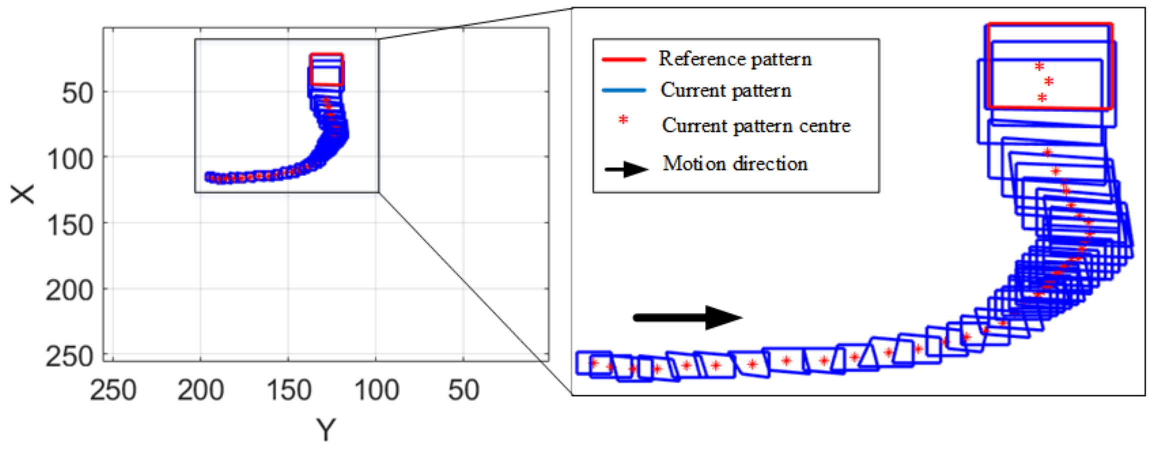


Figure 2.13: Pattern 'trajectory' in the image plane.

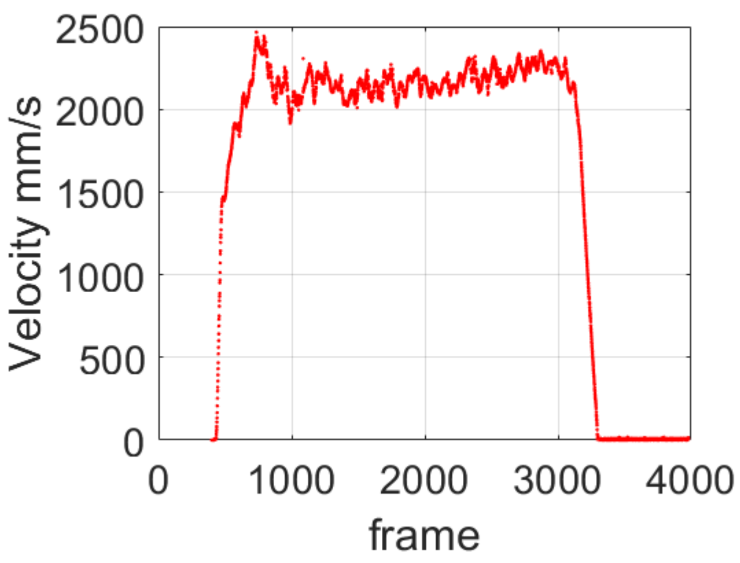


Figure 2.14: Velocity curve of the RC car during the whole navigation process.

igation speed is calculated 2.20 m/s and the maximum speed reaches 2.50 m/s according to the VICON tracking data, which shows the rover response to the environment quickly with the guidance of the SCAMP-5 vision system.

The time cost for each procedure of the whole control system, including that in SCAMP-5 and Raspberry Pi, is measured and listed in Table 2.2. Our SCAMP-5 pipeline is able to perform at over 2000 fps according to Table 2.2. This accounts for the image and PID processing but does not include the image exposure time which will depend on environmental illumination conditions.

To further explore the agility of the robot system based on the SCAMP-5, we then

| Processing Steps | Time Cost ($\mu s$) |
|---|---|
| Image capturing and thresholding | 52 |
| Flooding method to extract dots (including normal and broken squares) | 63 |
| Image de-noising | 4 |
| Getting coordinates of four dots' centre | 328 |
| Coordinates transmitted to Raspberry Pi | ≈ 24 |
| PID control in Raspberry Pi | ≈ 1 |
| Total | ≈ 472 |

Table 2.2: Time cost for different components of the system.

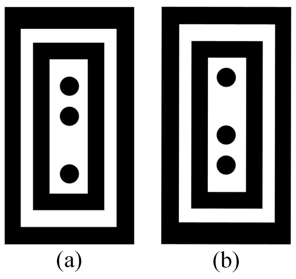


Figure 2.15: 'Slalom' pattern. (a) Turn right for certain degrees; (b) Turn left for certain degrees.

proposed a 'slalom' application with some pre-designed visual patterns (see Figure 2.15) encoding turning and angle information which are extracted using the SCAMP-5 vision system. By placing these patterns in specific positions, the rover weaves left and right around the patterns according to the information encoded on each, in a similar manner to slalom skiing. The image processing method is similar to the one used for four disks extraction.

Figure 2.16 shows the trajectory and velocity of the rover during navigation. In Figure 2.16(a), there are eight patterns placed in the arena, six of them guide the rover to turn right and two to turn left. The robot turns then continues towards the next closest pattern it encounters. Figure 2.16(b) shows the average speed of 1.70 m/s for this scenario. By placing these gates in a line, a high-speed slalom can be achieved. In Figure 2.16(c), the distance between gates are set 2.40 m apart, and the rover turns right or left at a distance 0.80 m from the pattern. Its velocity curve (see Figure 2.16(d)) shows the highest speed reaches 3.88 m/s which translates to 139 km/h for the real-sized car.

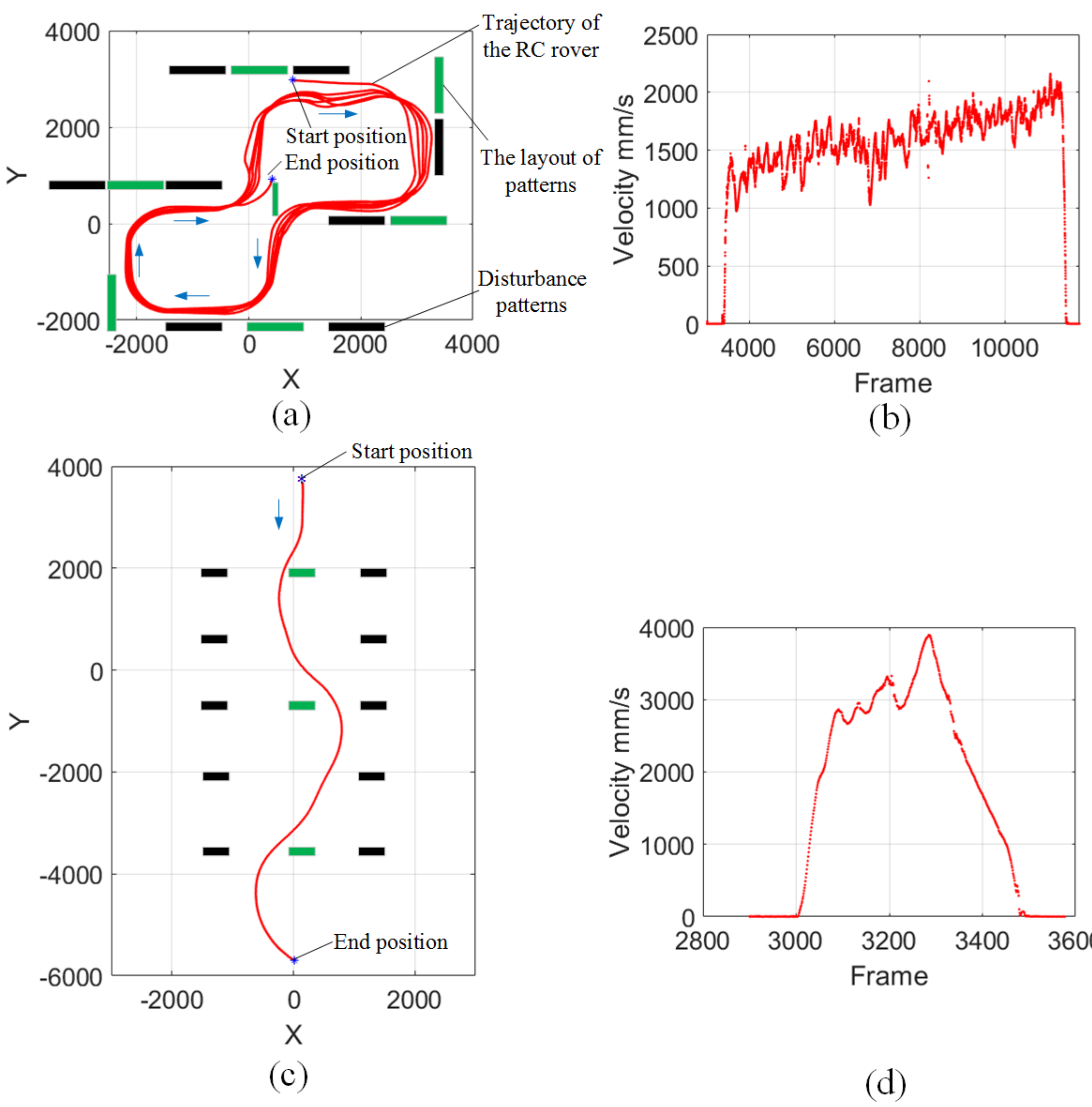


Figure 2.16: Trajectory and velocity curve for the 'slalom' process. (a) trajectory of number '8'; (b) velocity curve for number '8'; (c) trajectory of a fast 'slalom'; (d) velocity curve for the fast 'slalom'.

#### 2.2.5.1 Mobile Car Drifting

This section illustrates briefly the phenomena of a car sliding when it approaches a gate at high speed. If the rover is travelling at a fast rate of speed after passing one gate, drifting is a possibility when the rover changes direction to reach the next gate. Indeed, when the rover is sprinting toward the front gate, the following gate is invisible. When the rover's perspective of the next target changes quickly, the angle of the steerable

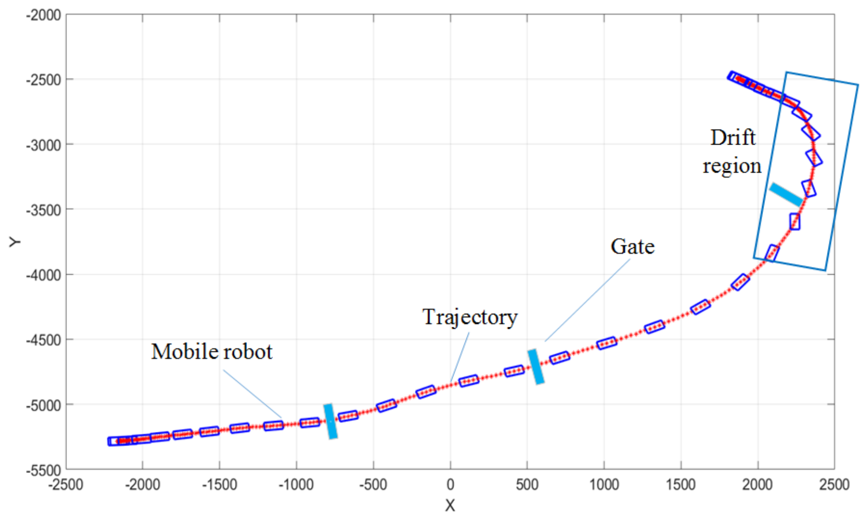


Figure 2.17: Trajectory of drifting during an agile reactive navigation.

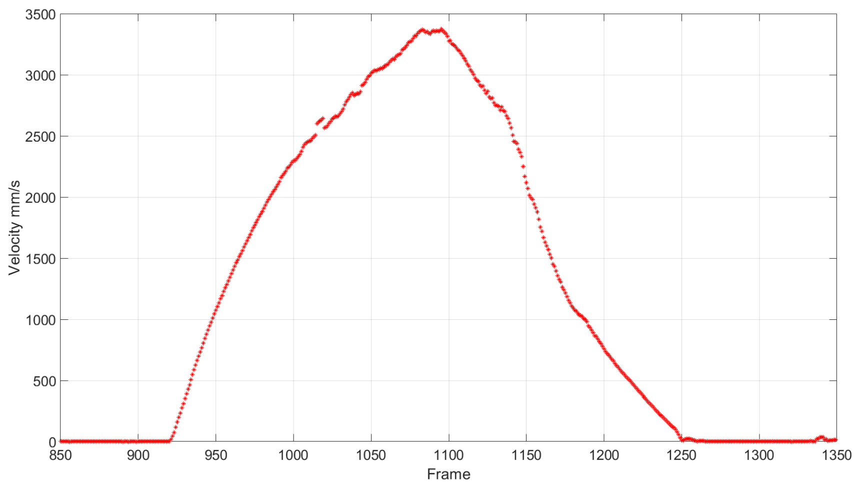


Figure 2.18: Speed curve during an agile reactive navigation.

wheel changes abruptly. It is impossible to regulate the rover's direction while drifting unless the target is always visible to the camera.

In Figure 2.17, the blue rectangle represents the position and orientation of the mobile robot, and the robot is speeding from the left to right. It is apparent that, at first, the robot's moving direction is same with the tangential direction of the trajectory. However, as the rover accelerates toward the the third gate, the car starts to skid and out of control. Finally, it is unable to traverse the third gate. In essence, slip occurs when lateral friction becomes smaller than centrifugal force, $f < F$. According to Newton's second law of motion $F = mv^2/r$. A slide is more likely to occur when the velocity is higher

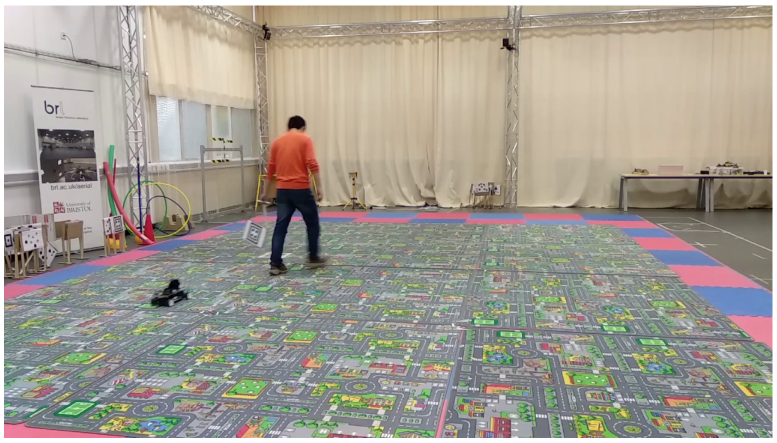


Figure 2.19: Dynamic pre-set pattern tracking with a SCAMP PPA.

and the turning radius is smaller. In this scenario, when the mobile robot gets sight of the third gate, it instructs to steer the front wheel, hence the turning radius becomes smaller suddenly resulting in the mobile robot's drift. The velocity curve is plotted in Figure 2.18. According to the recorded frames in Figure 2.18, the frame number is about 1140 when the mobile robot is going to slide. The corresponding velocity in Figure 2.18 is around 3.3 m/s. Notice that the maximum speed that results in sliding caused by many factors. Considering the different steering angle and ground contact situation, the average navigation speed 1.871 m/s and maximum navigation speed 3.1 m/s is possible the maximum navigation speed for this system configuration. It is worth noticing that drifting happens when the rover changes its direction rather than it reaches the highest speed.

**Dynamic Object Tracking:** We then demonstrate dynamic object tracking with a similar configuration to further show the robustness and potential applications. An experimenter moves the pattern randomly within the experiment arena, where the mobile robot localises the pattern and tracking without loosing it. Experimental video can be seen from `https://youtu.be/vvzJBpuuwLM`.

Lastly, we believe that the development of novel vision processing hardware architectures is key to the progress of agile and responsive robotics that are required to operate in a complex and uncertain world. This work explores the case of agile reactive navigation and our future emphasis will include extending the capabilities beyond known target detection and into classification and place recognition while navigating.

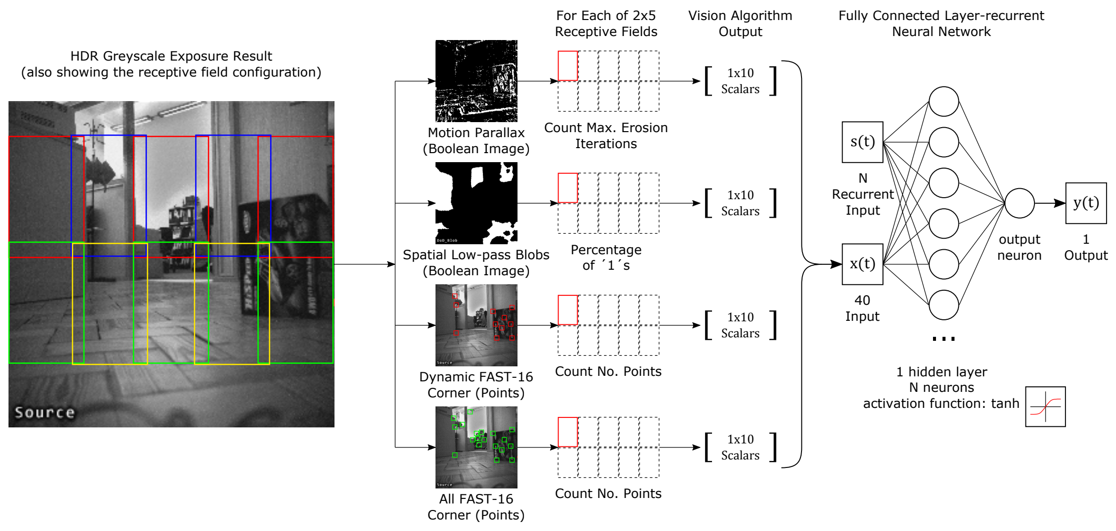

HDR Greyscale Exposure Result
(also showing the receptive field configuration)
Source
For Each of 2x5 Receptive Fields
Vision Algorithm Output
Fully Connected Layer-recurrent Neural Network
Motion Parallax (Boolean Image)
Count Max. Erosion Iterations
Spatial Low-pass Blobs (Boolean Image)
Percentage of ´1´s
Dynamic FAST-16 Corner (Points)
Count No. Points
All FAST-16 Corner (Points)
Count No. Points
1x10 Scalars
s(t)
N Recurrent Input
x(t)
40 Input
output neuron
y(t)
1 Output
1 hidden layer
N neurons
activation function: tanh

Figure 2.20: A diagrammatic representation of the vision-based proximity estimate system. The vision chip generates four distinct sorts of feature pictures. These photos are digested using Receptive Fields into a vector of 40 scalar values. On the MCU, the layer-recurrent neural network is used to simulate a proximity sensor by utilising just the input vector and the output of all neurons from the previous time step ($S(t)$). (Figure from [67]).

## 2.3 Robot Obstacle Avoidance via Proximity Estimation Using Vision Features Extracted on Sensor

This section is a joint work [115] as the second author, where the author contributed to parallel feature extraction, and robot free-roaming experiments.

Section 2.2 explores the robot agile reactive navigation with pre-set fixed patterns as the prior knowledge, which is hard to generalise to any indoor environment. In order to take advantage of the environment feature information without relying on specific objects, this section presents a monocular vision based proximity estimation system using abstract features, such as corner points, blobs and edges, as inputs to a neural network. Firstly, an experimental mobile platform was built by integrating the SCAMP-5 vision system, a micro-controller, and an RC model car. We present a series of feature extraction algorithms on the focal plane of the SCAMP PPA in a way that only a small amount of sparse descriptors is filtered. Then, a small recurrent neural network is trained using these descriptors and the groundtruth distance information collected from three infrared proximity sensors. Finally, reactive obstacle avoidance behaviour is enabled with our

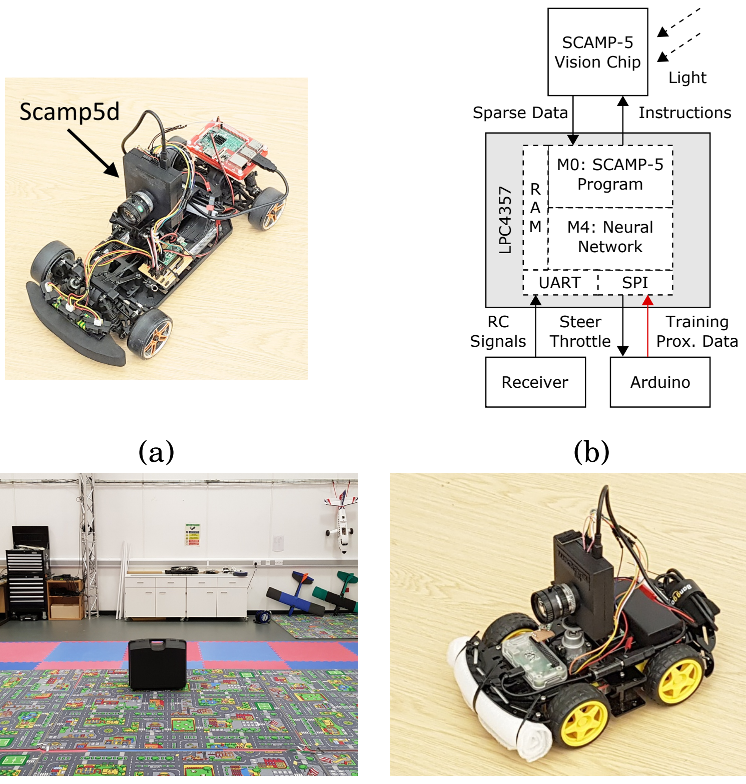


Figure 2.21: Hardware overview. (a) The RC car chassis for experiments.(b) The hardware block diagram. Experimental setup: (c) The environment of the systematic experiment and the obstacle, which was in a differently building and scenario. (d) The differential wheeled rover used in the experiment. The vision system and the proximity estimation neural networks on the rover were same as those on the RC model car. Only the collision avoidance controller was different, which enabled this vehicle to move fully autonomously. (Figure from [67]).

trained neural network for a mobile robot platform. The response frequency of the robot control system from sensing to network inference results is > 250 Hz enabling the small ground vehicle navigate indoors at speeds of 0.64 m/s to 1.8 m/s in the experiment.

In this work, three types of features that contain monocular depth cues were specifically investigated: (a) spatial frequency filtering, (b) motion parallax, and (c) corner features. The receptive field approach [135] was used to digest these feature pictures into a vector of scalars that could then be processed by a fully connected neural network (Figure 2.20), the output of which was the proximity distance between the user and any adjacent obstacles. With this technology, it is possible to take advantage of the efficiency of on-sensor and near-sensor parallel image processing hardware while still

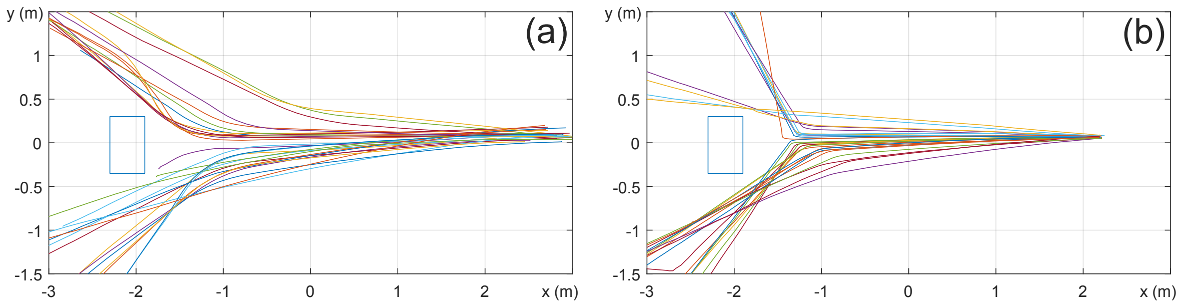


Figure 2.22: Vehicle trajectories during the obstacle avoidance experiment. The blue rectangle denotes the impediment. The speed of the vehicles in $x \in (-1.0, 1.0)$ were measured. (a) The RC model car. The reactive controller using the neural network output as the proximity input, and outputting the angle of the steering servo. The throttle was controlled manually. The minimum, maximum and average speed of the car were 0.90 m/s, 1.87 m/s and 1.15 m/s correspondingly. (b) The fully autonomous differential wheel vehicle. The reactive controller using the neural network output as the proximity input, and outputting the speed of the wheels. The minimum, maximum and average speed of the car were 0.64 m/s, 0.74 m/s and 0.71 m/s correspondingly. (Figure from [67])

maintaining the efficacy of the neural network-based approach to image processing. Using built-in functions of the SCAMP PPA, such as $Blur$, $EventReadout$,$GlobalOr$, and $GlobalSum$, a vector of values can be read out to the MCU from the results of image processing algorithms performed on the sensor. This thesis contributes to following feature extraction methods and experiments.

**Spatial frequency filtering:** It is usual (but not always) for a close item or surface on an image to appear as a uniformly coloured area with a low spatial frequency. A Difference of Gaussian (DoG) process can be used to extract patterns with a specified spatial frequency, as [136] demonstrates. The DoG-like function can be approximately implemented with function $Blur$. Then the blurred image is binarised to image with the percentage of white area as results for the following neural network.

**Motion parallax:** Motion parallax contains monocular depth cue when objects moving within a camera, which is to say a closer object moving relatively faster in the field view. Hence, wide edges more likely come from a closer object. We thus take advantage of the edge width information calculated by subtracting two consecutive frames and then obtain the maximum width. Details on motion parallax computation can be seen from Algorithm 8. Details on FAST-16 corners extraction can be seen from paper [137].

**Free Roaming Experiment:** To assess the system's performance in the complicated environments, the mobile were placed in an uncontrolled area (Figure 2.21) and allowed to travel freely while avoiding collisions (Figure 2.22) . The video of these trials can be found from https://youtu.be/UUrXKUTB8r8. Additionally, the video depicts several failure mechanisms. For example, the proximity sensors utilised in the training process have difficulty detecting chair legs, particularly swivel chairs. As a result, the trained vision-based system was unable to recognise such structures. Additionally, the vision-based estimating method failed to detect impediments that were intrinsically difficult to see from the camera's perspective, such as low steps on the floor.

## 2.4 Weighted Node Mapping and Localisation

This section is based on a collaborative work of paper [7] as the second author, where the author contributed to Local Binary Pattern based mapping and localisation. Earlier sections examined reactive navigation by mobile robots using either pre-set patterns or global environmental information. This section further explores on-sensor topological 1-dimensional mapping and localisation which is potential to be utilised in robotics.

Our methods predict the correct node inside a topological map generated from an image sequence by analysing image similarity, spatial coherence and taking advantage of the PPA's parallel nature. Our implementation operates at a frequency of +300 Hz on big public datasets with +2K locations, consuming 2.5W at a rate of 500 GOPS/W. We compare our method to more conventional ways and find that our method outperforms them on F-1 performance, even in simulation. As far as we are aware, this is the first on-sensor mapping and localisation solution.

### 2.4.1 A 1-D weighted node-map algorithm for visual route recognition

The topological map, $\boldsymbol{X}_t$, in this work contains $N$ nodes that correspond to $N$ labelled locations. We compare two techniques to place representation: local binary descriptors (8×4 pixels) and low-resolution pictures (8×8 pixels) with same mapping and localisation algorithms. Thus each node is described by either a binary descriptor or a scaled-down image, which are stored as a 'database' on the SCAMP PPA's focal plane. Additionally, such nodes are spatially coherent, and each one can be associated with GPS coordinates. A set of nodes $x_t^{[t]}$ at the time $t$: $\boldsymbol{X}_t = x_t^{[1]}, x_t^{[2]}, ....., x_t^{[N]}$, where $N$ is the total number of nodes. Each of them has an associated weight value ${w_t}^{[i]}$, $\boldsymbol{W}_t$ represents the set of weights for the set of nodes $\boldsymbol{X}_t$. .

**Motion model estimation**

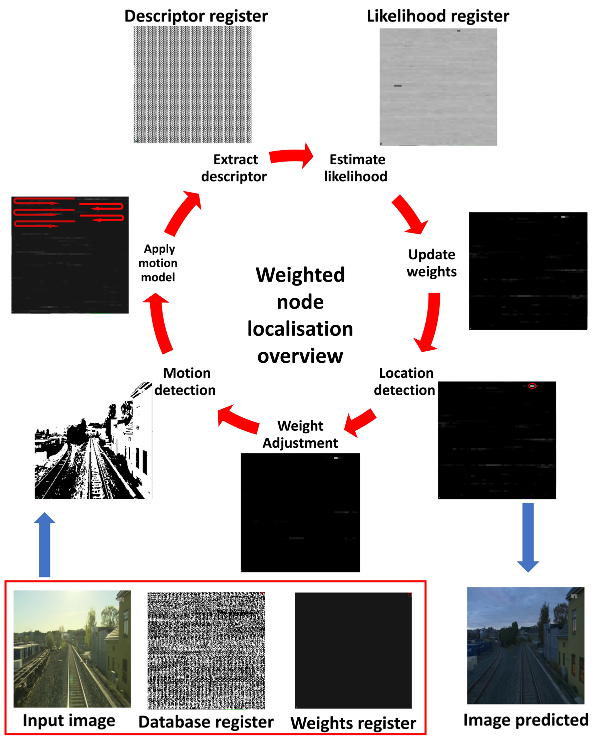
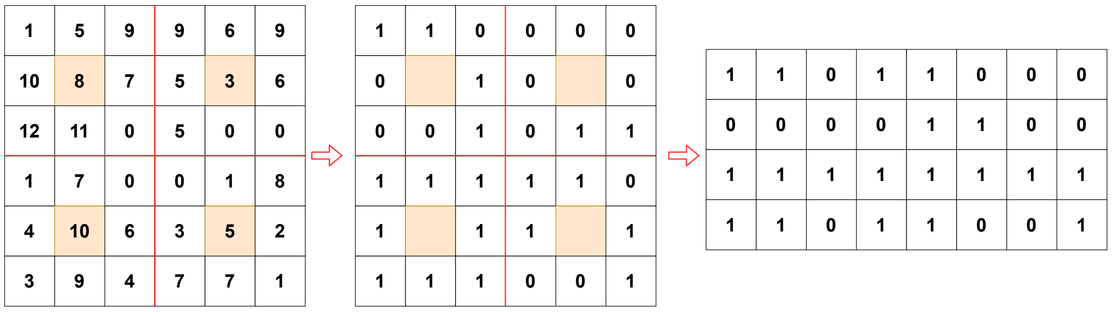


Figure 2.23: Overview of the iterative weighted-node process using binary descriptor method (Figure from [7]).



| | | | | | |
|---|---|---|---|---|---|
| 1 | 5 | 9 | 9 | 6 | 9 |
| 10 | 8 | 7 | 5 | 3 | 6 |
| 12 | 11 | 0 | 5 | 0 | 0 |
| 1 | 7 | 0 | 0 | 1 | 8 |
| 4 | 10 | 6 | 3 | 5 | 2 |
| 3 | 9 | 4 | 7 | 7 | 1 |

| | | | | | |
|---|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | | 1 | 0 | | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 |
| 1 | | 1 | 1 | | 1 |
| 1 | 1 | 1 | 0 | 0 | 1 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |

Figure 2.24: The LBP descriptor extraction process.

The first step of our proposed method is initialisation on both database and register weights and then combines two different hypotheses for localisation: image similarities and a motion model. This one-dimensional technique makes use of the motion model in the following manner: forward movement is associated with the notion that each weight is travelling to the next node in the topological map. The nodes and their associated weights are successively stored in the PE array, beginning with the rightmost upper corner; thus, progressing entails shifting to the left and down.

**Temporal weight value estimation**

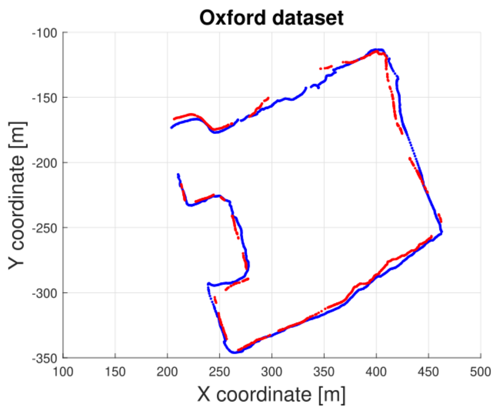
The motion model has been calculated and applied to the collection of nodes $\boldsymbol{X}_t$ at this point. The following step is to estimate the temporal weights associated with the likelihood of the current input measurement ($z_t$, either low-resolution image or binary descriptor approach) being at that node in the topological map. The image pre-processing approach for Local Binary Pattern method: Hamming distance (Equation 2.8) with 32-bit local binary descriptors which is utilised to calculate the distance between the current input image and all those in the database [138]. Taking advantage of the powerful parallel computing capability in the SCAMP PPA, these comparison operations can all be performed efficiently. To achieve this on-sensor parallel computing, the image should be placed in a specific way: consider the case where image descriptors have the dimension of $4 \times 8$ pixels; it will occupy 32 pixels (Figure 2.24). Given the *SCAMP-5d* resolution, $256 \times 256$, a total of $64 \times 32$ images of $4 \times 8$ can be processed simultaneously in that DREG. This implies that with this size of images, the maximum number of nodes $N$ is 2048 on one DREG. Notice that the database must be loaded in the same way the images are stored, using the *snake* stack pattern. At every iteration, when a new greyscale image is captured, it is transformed to $4 \times 8$. This is done by applying local binary pattern method, as shown in Figure 2.24, which is a simple but efficient feature operation method by binarizing the difference between the pivotal pixel and its neighbour pixels as the input for later processing [139]. The database is created using the *snake* pattern in a different DREG. Therefore, the input image could be compared with the entire database by performing a subtraction followed by an absolute operation between database and input image DREG (both after LBP extraction).

$$likelihood(z_t|x_t^{[i]}) = HammingDistance = d(I_{input} - I_{db}) \tag{2.8}$$

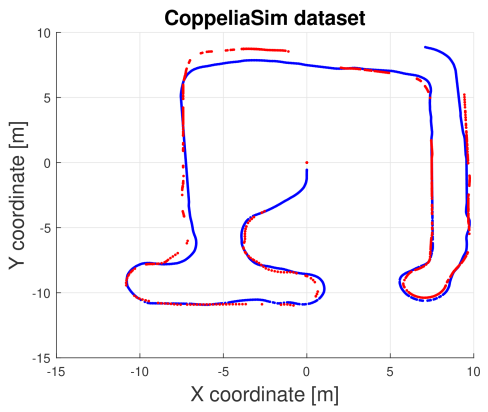
where $I_{input}$ is the input image; $I_{db}$ represents all images in the database. A $HammingDistance$ is obtained for each block, which then stored in AREG representing node's weight. The algorithms can be seen from Appendix A.2.

**Updating node weight**

The node weights values are updated based on the motion model and the temporal weight estimation. The node weight shifting applies the motion model described earlier, and the temporal weights at this point have been already calculated. $\alpha$ is introduced to estimate the new weights values. Likelihood is represented by image similarity obtained from Hamming distance. Hence, the smaller the value, the more similarity between two images. Equation, $w_t^{[i]} = w_{t-1}^{[i]} + (\alpha - \mu_t^{[i]})$, transforms the temporal weight values to a better representation.

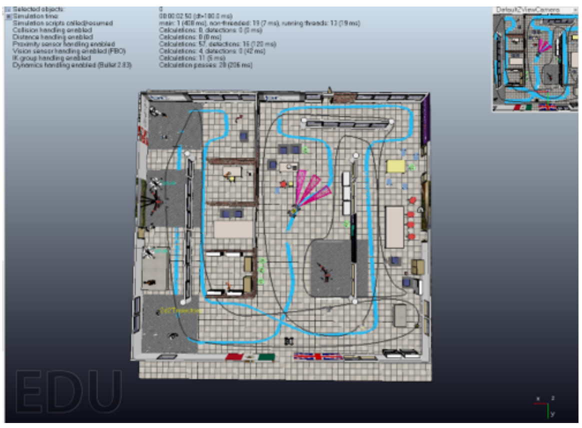


(a)

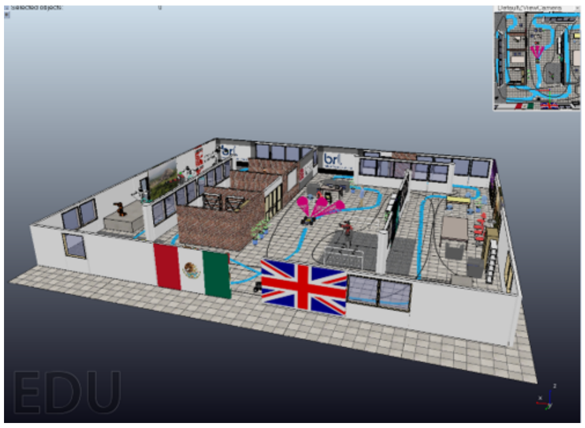


(b)

Figure 2.25: (a) Position estimated by our proposed low-resolution images approach implemented on *SCAMP-5d* with Oxford dataset. (b) Position estimated by binary descriptor approach when implemented in the *SCAMP-5d* testing with images generated by the simulated environment. Blue points represent the locations of the input image and red points the correctly predicted locations. (Figure from [7]).



(a)



(b)

Figure 2.26: (a) Top view of the *scene* in CoppeliaSim which simulates a laboratory environment. (b) Front view of the laboratory scene. (Figure from [7])

### Location detection, weight adjustment, and mapping

After each estimation iteration, the result must be verified by scanning all weights to find the one with the highest value associated with the correct position. Rather than relying on a single image, a sequence-based method is used in this work. To prevent saturation of the AREG storing the weights, we adjust them after every sequence of $k$ images as follows. The highest weight value is multiplied by a factor $\gamma$, which is within $[0-1]$; this value is then subtracted from the weights. A new set of node weights is

Table 2.3: F1-Scores and localisation time evaluation.

| Dataset | Baseline | Our simulation | Scamp-5d system |
|---|---|---|---|
| Norland | ABLE-M[142] 97% | Binary Descriptor 99.57%<br>Low resolution images 98.16% | Binary descriptors 93.38%<br>Low_resolution images 71.78% |
| | 9.3 ms | | |
| | SeqSLAM[144] 92% | | 3.168 ms |
| Oxford | Liu & Zhang[143] 93.04% | Binary Descriptor 98.98%<br>Low-resolution images 97.84% | Binary descriptor 64.95%<br>Low-resolution images 80.4% |
| | 800 ms | | 2.59 ms |
| Coppeliasim | ———- | Binary Descriptor 86.13%<br>Low-resolution images 95.03% | Binary descriptor 82.7%<br>Low-resolution images 82.6% |
| | | | 3.168 ms |

created by applying equations $w_t^{[i]} = w_t^{[i]} - \gamma max(\boldsymbol{W}_t)$ and $\boldsymbol{X}_t = \boldsymbol{X}_t + (\boldsymbol{W}_t)$. Mapping in our approach takes the form of database creation from a sequence of input images. 32 bit binary descriptors (Figure 2.24) are stored in a DREG in a 'snake' fashion (Figure 2.23).

### 2.4.2 Experimental Results

This section shows the results of our implementation previously presented. We perform experiments across several publicly available datasets including Nordland dataset [140], Oxford dataset [141], and a simulated laboratory environment. F1 score[4] is adopted to quantify the performance of the localisation methods proposed above. Localisation tasks are shown across three datasets, two of which are from outdoor environments and one from a simulated laboratory on a mobile robot. Table 2.3 shows a comparison of F1 scores across the different implementations and datasets. This measurement is used to combine precision and recall information into one value which describes the accuracy of the system. Our fully parallel algorithm takes 3.17 milliseconds (ms) (315 fps) for the local binary pattern method and 2.59 ms (386 fps) for the low-resolution image method respectively. Note that other methods such as [142], [143] deployed their systems on powerful external conventional computers. But while the PPA concept and hardware is still in development, its fundamental parallel nature surpasses their performance, for current hardware requires low power consumption ($\leq$ 2.5W) and performs most computations on-sensor. More experimental results can be seen from Appendix A.4.

## 2.5 Conclusion

This chapter illustrates the proposed various image processing algorithms on the sensor for multiple tasks, including pre-set pattern extraction, feature extractions for RC car

[4]https://en.wikipedia.org/wiki/F-score

obstacle avoidance, and on-sensor visual feature extraction for mapping and localisation.

Firstly, this work focuses on real-time visual information processing using a Pixel Processor Array and agile robot navigation with information derived from the visual sensor. The SCAMP-5 is capable of processing high-frame rate visual information as the agile navigation requires. In this section, we implemented the high-speed reactive navigation and enable a Subaru mobile robot to run through multiple gates at an average speed of 2.20 m/s and a highest speed of 3.88 m/s for a 'slalom'. We proposed a direct link between the image information and the angle of the steerable servo motor in the mobile robot. To the best of our knowledge, the speed of the agile ground vehicle achieved is the highest among the systems using non-conventional visual-processor pipelines. The multi-gate test shows high-speed navigation for an agile reactive navigation system and shows the effectiveness of the control method under motion disturbance and vibrations. However, the navigation speed and performance is still limited by many factors including low control frequency of the servo motor or loss of targets especially when the fast-moving mobile robot is far from the target.

Secondly, we introduce a new parallel node weighting approach for visual route mapping and localisation using a dynamically programmable SCAMP PPA sensor-processor. As far as we are aware, this is the first example of on-sensor mapping and localisation. By embedding a dataset in the registers of the sensor's processing array, our approach enables the majority of computation to be performed efficiently in parallel on the sensor's focus plane. We introduced a map representation, in particular, local binary descriptors. The results were demonstrated using publicly available datasets. Due to the prototype nature of the PPA hardware deployed, we still observe a performance gap between simulation and real-world PPA hardware due to implementation constraints. Nonetheless, our node-weighted localisation algorithms outperform baseline techniques such as SeqSLAM and ABLE-M, demonstrating the promise of such on-sensor processing. Our objective is to show that this work sheds light on an emerging trend of unique visual architectures that efficiently combine perception and computing, hence enabling new potential for complicated and agile robotic tasks.

Lastly, we conceived and constructed a monocular vision-based proximity estimate method utilising the Scamp5d edge vision system. On the focus plane, the vision chip collected the image and used three different types of feature extraction techniques. The vision chip's only output was scalar data processed from the visual characteristics. Similar to proximity sensors, the MCU in the vision system then uses three layer recurrent neural networks to assess the proximity distance. When the automobile was driven manually, training data for the neural network were collected. The obstacle avoidance experimental trials established the system's overall efficacy. The system's power consumption (< 2.1 W) and latency (< 4 ms) are low because: (a) the feature extraction algorithms take advantage of the vision chip's hardware acceleration and can thus be executed extremely efficiently without requiring image transfer; and (b) the neural networks running on the MCU operate on abstract feature vectors and are thus relatively simple.

However, all these image processing algorithms are designed for specific tasks, limiting on-sensor machine vision to a broader range. The essential method used for these three applications is feature extraction. The author has designed several different ways to extract useful information and fuse this information for different tasks. However, feature extraction from raw data should be carefully designed, which would be unsuitable for more sophisticated tasks, especially when combining many different types of features are needed. With this in mind, the following chapters will focus on the emerging and more general methods for higher-level image processing: convolutional neural network (CNN). With on-sensor CNN inference, sensors can deal with classification, regression, and segmentation in a more general way. Hence, kernel filers for feature extraction can be learned automatically during the training process.

CHAPTER 3

# ON-SENSOR HIGH-SPEED LIGHT-WEIGHT BINARY CNN INFERENCE

Earlier chapters focus on specific low-level image features and their fusion for robotic reactive scenario, navigation, and localisation. Although hand-crafted parallel image processing methods on the SCAMP PPA are efficient and sufficient for the given tasks, they are limited to a particular environment, hence challenging to generalise to a broader range of applications. Therefore, in the following chapters, we start to explore learning-based image feature extraction and combination with on-sensor quantised neural network to expand the on-sensor signal processing to more general visual inference tasks, such as classification and coarse segmentation. The contributions of this chapter are published in [65].

## 3.1 Introduction

Performance, storage, and power consumption are three major factors that restrict the use of machine learning algorithms on embedded systems. However, new hardware architectures designed with visual computation in mind may hold the key to solving these bottlenecks. This work makes use of an emerging novel visual sensor device: the pixel processor array (PPA), to embed a convolutional neural network (CNN) onto the focal plane. We present a new high-speed implementation of strided convolutions using binary weights for the CNN on PPA devices, allowing all multiplications to be replaced by more efficient addition/subtraction operations. Image convolutions, Rectified Linear Unit (ReLU) activation functions, max-pooling and a fully-connected layer are all performed directly on the PPA's imaging plane using the analogue current signals, exploiting its massive parallel computing capabilities. The author demonstrates CNN inference across four different applications, running between 2,000 and 17,500 fps with power consumption lower than 1.5 $W$. These tasks include identifying 8 classes of plankton, hand gesture classification and digit recognition.

Convolutional neural networks (CNN) already play a significant role in modern computer vision tasks, such as classification, localisation, and segmentation. With the ever-increasing prevalence of mobile and embedded devices, such as smartphones and mobile robots, there is strong motivation to enable CNNs on portable lightweight devices [145–147]. However, state-of-the-art CNN-based methods are typically heavily GPU reliant, and difficult to deploy on the embedded systems without optimisation or modification [148]. The Three main issues associated with this are the lack of parallel computation power, memory, and battery life, all of which are required by computationally demanding CNN algorithms. Two potential solutions are (1) hardware acceleration [149–151] and (2) data compression in terms of storage and complexity using techniques, such as network pruning and low-bit quantization of network weights [152, 153].

Rather than using a conventional approach in which a camera streams video frames to processing hardware, this paper focuses on implementing CNNs upon a novel, general-purpose, Pixel Processor Array (PPA) (Figure 1.2). Our approach takes advantage of the PPAs massively parallel architecture to efficiently execute a binary CNN. Image convolutions, activation functions, max-pooling and fully-connected layer are implemented upon the PPA. By adopting an "in-pixel" weight approach such as [64], our implementation is significantly faster than many existing works [63, 113, 114] and does not rely on external processing. Training is performed offline upon a standard PC while inference experiments are performed entirely upon the PPA. This work seeks to illustrate the potential high-speed CNN applications that can be achieved upon such PPA devices.

The main contents of this section are: (i) A new image convolution implementation for PPAs, incorporating variable convolution stride to allow for faster inference times compared to previous works [59, 114], increasing the inference speed across various tasks depending upon the task's level of complexity. (ii) The demonstration of our fast SCAMP-5 CNN implementation across a wider and more complex set of tasks than previous works, which had predominately focused upon only demonstrating MNIST classification. We demonstrate real-time hand gesture recognition, plankton classification from the

National Data Science Bowl plankton dataset [154] along with digit recognition. PPA inference speed for our approach is extremely fast across all tasks, ranging from 2000 to 17500 fps.

## 3.2 Neural Networks with Other Emerging Sensors

Section 1.2.2 introduced on-device computing sensors in terms of hardware. Alternatively, to enable high-performance CNN inference on embedded devices, a great amount of work has been carried out on hardware accelerators and unconventional visual sensors.

### 3.2.1 Hardware Accelerations

The on-going work on implementing hardware accelerators for the efficient execution of CNN on edge devices has resulted in numerous architectures and prototypes proposed in recent years by academic groups, for example [16, 150, 151, 155–157], as well as commercially available neural network accelerator IP blocks [158] or dedicated hardware devices [159, 160]. The need for the co-optimisation of the architecture, from image sensor, through image signal processing, to neural network acceleration is recognised as an important aspect of vision system design for embedded systems [161].

### 3.2.2 Unconventional Visual Sensors

Recent works using unconventional visual devices for CNNs have mainly focused on Dynamic Visual Sensors (DVS), SCAMP PPAs and other emerging intelligent sensors. DVS sensors produce data in the form of sparse contrast-change events, that facilitate low-latency visual processing using external computational hardware [29–31]. Then, these pre-processed binary events need to be transferred to processing units to carry out the inference using spiking neural networks. However, it is still challenging to perform CNN with a DVS considering the output features of sparse event points [32]. In addition, the development of image sensors equipped with embedded processing technologies, such as the Sony IMX500 sensor, enables on-device processing, where a DSP is dedicated to neural network signal processing. This on-device processing is preferable in terms of communication burden reduction [25]. Aistorm Mantis features charge domain visual and audio processing [26] with embedded neural network. Some simple demonstrations have been released including hand gesture classifier, person classifier, and key word spotting. There have been a number of studies using Eye-RIS vision system, including

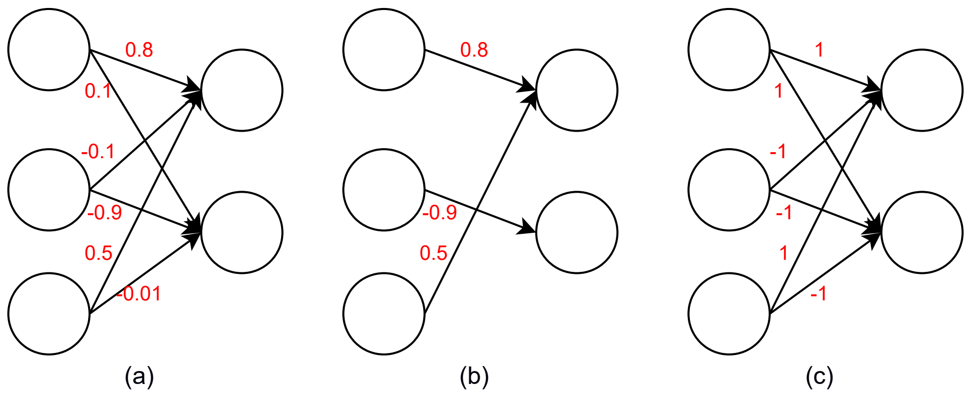


Figure 3.1: Schematic diagram of network compression scheme. (a) full-precision neural network. (b): pruned neural network. (c): binary neural network.

Data Matrix Code Recognition [162], moving target tracking with a mobile robot [163], and Cellular Neural Network control algorithms [164]. However, the most related neural network research is [165] where Eye-RIS served as a pre-processor for a ConvNet. A memory resistor (memristor) is fundamental electrical components and the resistance of memristor can be used to mimic the synaptic connections among neurons [28]. Hence the memristor is only in charge of computing.

As far as we know, there are no visual sensors that perform neural network inference fully on the focal plane within these above-mentioned related works, which means external hardware assisting the inference is essential and extra information transmission is necessary. However, this thesis offered a number of strategies for completely deploying neural networks over the SCAMP PPA's focus plane with the needs of external processing units.

## 3.3 Binary CNN Training for the PPA

### 3.3.1 Neural Network Compression

Although neural networks have proved to be highly effectively in a wide range of applications, the model size of a neural network is typically large with a great number of float-point weights and biases resulting from many convolutional and fully-connected layers, which needs high computational and memory costs [166]. As a consequence, neural networks are not friendly to the embedded and portable devices with limited computational power and stringent storage space. Hence, people are motivated to compress the full-precision neural network with various methods including neural network pruning [167] (Figure 3.1 (b)) and quantization [85, 168] to accelerate neural network and/or embedded neural networks into portable devices.

With so many neural network compression methods, this thesis takes advantage of neural network binary quantisation (Figure 3.1 (c)) to simplify the CNN on the focal plane of the PPA considering the small storage requirement of the binary weights and its feasibility to parallel implementation results from the binary weights. Binary neural network is an extreme example of network quantization, where the weights are trained to be -1 and 1. BinaryConnect [89] is one of the earliest research to compress the neural network by training binary weights.

### 3.3.2 Binary Neural Networks

There are mainly two processes to train a binary neural network: forward propagation and backward propagation. The following context illustrates how to obtain the binary weights through the training processes.

**Forward propagation**

Binarisation function converts weights to either -1 or 1 during forward propagation in the binary neural networks. Consequently, the comparatively time-consuming float-point-based multiplication operations can be replaced by either additions or subtractions, which is suitable for the SCAMP vision systems. There are two types of binarisation functions: deterministic and stochastic binarisation.

$$w_b = \begin{cases} +1 & w >= 0, \\ -1 & otherwise \end{cases} \tag{3.1}$$

Where $w_b$ is the binarized weight and $w$ is the full-precision weight. It is obvious to see the binarization method of Equation 3.1 is efficient and straightforward. The alternative binarization scheme is stochastic binarization, where weights are binarized with a probability.

$$w_b = \begin{cases} +1 & \text{with probability } p = \sigma(w) \\ -1 & \text{with probability } 1 - p. \end{cases} \tag{3.2}$$

where $\sigma$ is the "hard sigmoid" function [90]:

**Algorithm 3:** Binary neural network training. [89]

```
1  $C$ is the cost function, $L$ is the number of layers, and $a$ is the activations.
2  **Inputs**:
3  a mini batch of inputs and targets
4  previous parameters weights $w_{t-1}$ and biases $b_{t-1}$, and learning rate $\eta$
5  **Methods**: update parameters weights $w_t$ and biases $b_t$
6  **1. Forward propagation**:
7  $w_b \Leftarrow binarize(w_{t-1})$
8  For $k = 1$ to $L$, compute $a_k$ knowing $a_{k-1}$, $w_b$ and $b_{t-1}$
9  **2. Backward propagation**:
10 calculate the activation gradient of output layer $\frac{\partial C}{\partial a_L}$
11 For $k = L$ to 2, compute $\frac{\partial C}{\partial a_{k-1}}$ knowing $\frac{\partial C}{\partial a_k}$ and $w_b$
12 **3. Parameter update**:
13 Calculate $\frac{\partial C}{\partial w_b}$ and $\frac{\partial C}{\partial b_{t-1}}$ knowing $\frac{\partial C}{\partial a_k}$ and $a_{k-1}$
14 $w_t \Leftarrow w_{t-1} - \eta \frac{\partial C}{\partial w_b}$
15 $b_t \Leftarrow b_{t-1} - \eta \frac{\partial C}{\partial b_{t-1}}$
```

$$\sigma(w) = clip(\frac{w+1}{2}, 0, 1) = max(0, min(1, \frac{w+1}{2})) \tag{3.3}$$

The reason here to use the hard sigmoid rather than the standard sigmoid function $\frac{1}{1+e^{-x}}$ is because Equation 3.3 is far more efficient than its standard counterpart while obtaining a good approximation to the sigmoid. This thesis uses the former function for a more efficient training.

**Backward propagation**

Backward propagation's objective is to compute the cost function $C$'s partial derivatives $\frac{\partial C}{\partial w}$ and $\frac{\partial C}{\partial b}$ with respect to weights $w$ and bias $b$. The weights update is based on the Stochastic Gradient Descent (SGD) during backward propagation. The key idea to train the binary neural network is to binarize the weights in the forward propagation and update current floating-point weights based on the last full-precision weights and calculated gradients as shown in Algorithm 3. Notice that in Algorithm 3 Line 14, we update real-valued weights $w_t$ with computed gradient and last real-valued weights $w_{t-1}$, which is the same for the bias.

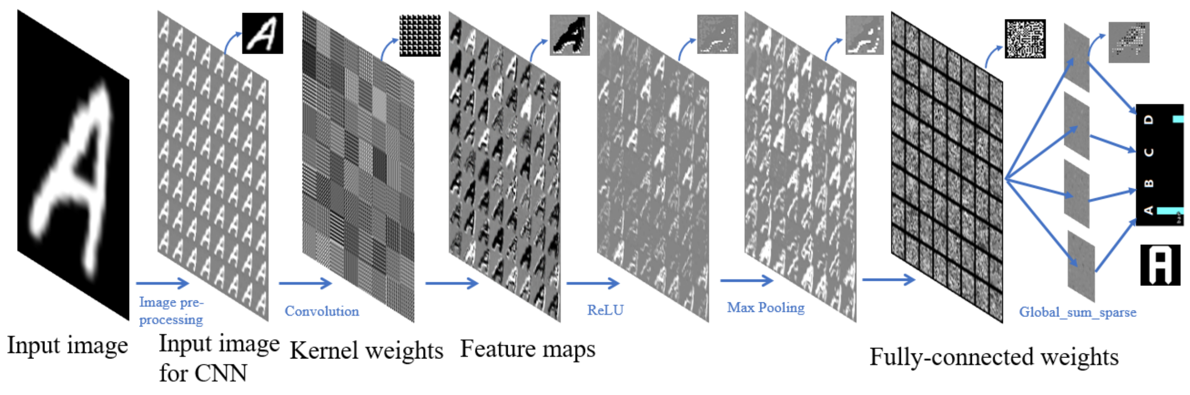


Figure 3.2: Parallel inference process by combining different registers and operations.

## 3.4 Binary CNN Implementation Method on the PPA

To achieve high-speed CNN inference, both the computation and weight-storage should be contained within the PEs of the processing array itself to fully exploit the PPA's parallelism and minimise data transfers. To this end, it is necessary to find a way to train the CNN with binary weights that can fit entirely within the PPA's array. This section describes the network training and implementation of high-speed CNNs for the SCAMP-5d PPA.

### 3.4.1 CNN with binary weights

In our work, the BinaryConnect scheme [89] is adopted and used to train binary weight networks. This produces simplified binary neural networks, whose weights can be stored entirely within the memory registers of the PPA array, but which still achieves acceptable accuracy. Additionally these binary networks are trained without neuron bias, further simplifying the CNN implementation [169].

This training scheme generates 1-bit weights representing values $\{-1, 1\}$ for both convolutional layers and fully connected layers. This allows rapid inference of various CNN layers to be performed using only native PPA arithmetic operations (additions/subtractions). The weights for convolutional and fully connected layers are directly stored in 1-bit digital registers on the array. This in-pixel weight approach first proposed in [64] allows for parallel and efficient implementation of CNN layers compared to methods which sequentially read weights from the controller [63, 113, 114].

Figure 3.2 shows the inference process of a CNN on SCAMP-5, with each step executed upon the image plane. First, input images are uploaded or directly captured into the

PEs of the array. To execute many convolution filters in parallel, this input image is pre-processed at runtime on the array, being down-scaled and then replicated to fill all 256×256 processing elements. In Figure 3.2 the input image is shrunk to 32×32 and replicated 64 times across the array. Each replicated image is associated with a different kernel filter, with 64 kernel filters arranged in-line with the 64 replicated image blocks. From this the convolutional layer generates 64 feature maps in parallel, followed by parallel activation function (ReLU) and max-pooling. Weights for the fully-connected layer are stored upon digital registers similar to that of the convolutional layer and are multiplied in parallel with their associated activation data. Finally, approximated sums of all pixels associated with each label are calculated by using 'sparse global summation' on the SCAMP-5 array, with the largest resulting sum representing the CNN's understanding of the image.

### 3.4.2 Convolutional Layer Implementation on-Sensor

This section implements the image convolution in a way that takes full advantage of the speed offered by the PPA parallel processing resources. Each kernel filter is replicated to the size of each input image block (Figure 3.2). Then the source image is "multiplied" by the corresponding kernel filters coefficients (+1 or -1) in parallel, with the convolution result obtained by the summation of pixels in the filter block. Moreover, strided convolutions (i.e. stride 1, 2, or 4) can be applied here for different applications to speedup inference process. This method allows the convolutional layer to be performed entirely on the PPA array using only native addition, subtraction, and image shifting operations.

Referring to Figure 3.4, 4×4 binary kernel filters for the convolutional layer are stored in 4×4 PE blocks using digital registers. Efficient multiplication of stored data by these binary weights can then be performed. The detailed layout of the 4×4 kernel filters is illustrated in Figure 3.3, showing how each of the 64 kernels is replicated multiple times to fill the 32×32 block of PEs holding the image it will operate on. Following the result of image multiplication, image convolutions (of stride 4) on the PPA are calculated by iteratively performing image shifting and addition a total of 12 times. As shown in Figure 3.5, the convolution results are stored in the bottom right corner of each 4×4 block. Convolutions of stride 1 and 2 can be calculated by simply repeating this process for stride 4 multiple times (×16 for stride 1, ×4 for stride 2. The second and third rows in Figure 3.5) illustrate this, using a different shifted copy of the kernel filter for each iteration.

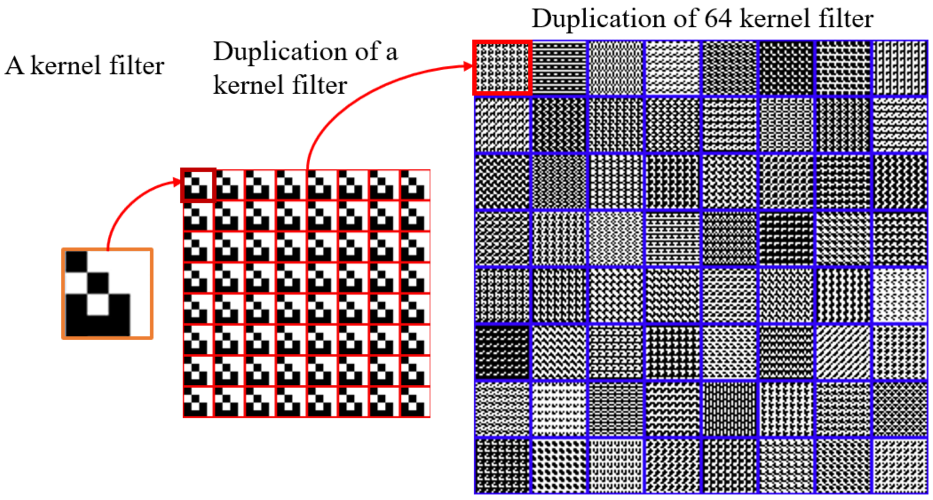

A kernel filter
Duplication of a kernel filter
Duplication of 64 kernel filter

Figure 3.3: The layout of 64 binary kernel filters in a digital register. Each filter can extract corresponding features from the initial input images to the downstream layers.

$$\begin{bmatrix} 1 & 2 & 4 & 7 & 9 & 8 & 6 & 3 \\ 5 & 5 & 6 & 8 & 4 & 8 & 7 & 9 \\ 1 & 2 & 6 & 8 & 9 & 6 & 1 & 7 \\ 2 & 3 & 8 & 2 & 6 & 0 & 4 & 9 \\ 7 & 7 & 8 & 2 & 9 & 1 & 8 & 0 \\ 7 & 2 & 3 & 5 & 3 & 6 & 3 & 8 \\ 1 & 4 & 3 & 3 & 5 & 5 & 0 & 8 \\ 3 & 4 & 5 & 4 & 5 & 1 & 7 & 1 \end{bmatrix} \times \begin{bmatrix} 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 & 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & -1 & 1 & -1 & -1 & 1 \\ 1 & 1 & -1 & -1 & 1 & -1 & -1 & 1 \\ 1 & -1 & -1 & -1 & 1 & 1 & 1 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & -4 & -7 & 9 & 8 & 6 & 3 \\ 5 & 5 & -6 & -8 & 4 & 8 & 7 & 9 \\ 1 & 2 & -6 & -8 & -9 & -6 & -1 & -7 \\ 2 & 3 & -8 & -2 & -6 & 0 & -4 & -9 \\ 7 & 7 & 8 & 2 & 9 & 1 & 8 & 0 \\ 7 & 2 & 3 & -5 & 3 & -6 & -3 & 8 \\ 1 & 4 & -3 & -3 & 5 & -5 & 0 & 8 \\ 3 & -4 & -5 & -4 & 5 & 1 & 7 & 1 \end{bmatrix}$$

Part of Source Image | 4 Kernel Filters | Result after parallel multiplication

Figure 3.4: The parallel implementation of multiplication. Each pixel of source image either remains unchanged or becomes negative according to the binary weights stored directly in registers.

It should be noted, for each iteration, only one pixel out of 4×4 block stores the correct value for image convolution. Hence, some degree of power efficiency is sacrificed compared to calculating 16 valid convolutional results for once. Despite this, even at stride 1 our implementation is still significantly faster at performing convolutional layers than many previous works [63, 113, 114] as multiple convolutional filters are executed in parallel across the array rather than sequentially.

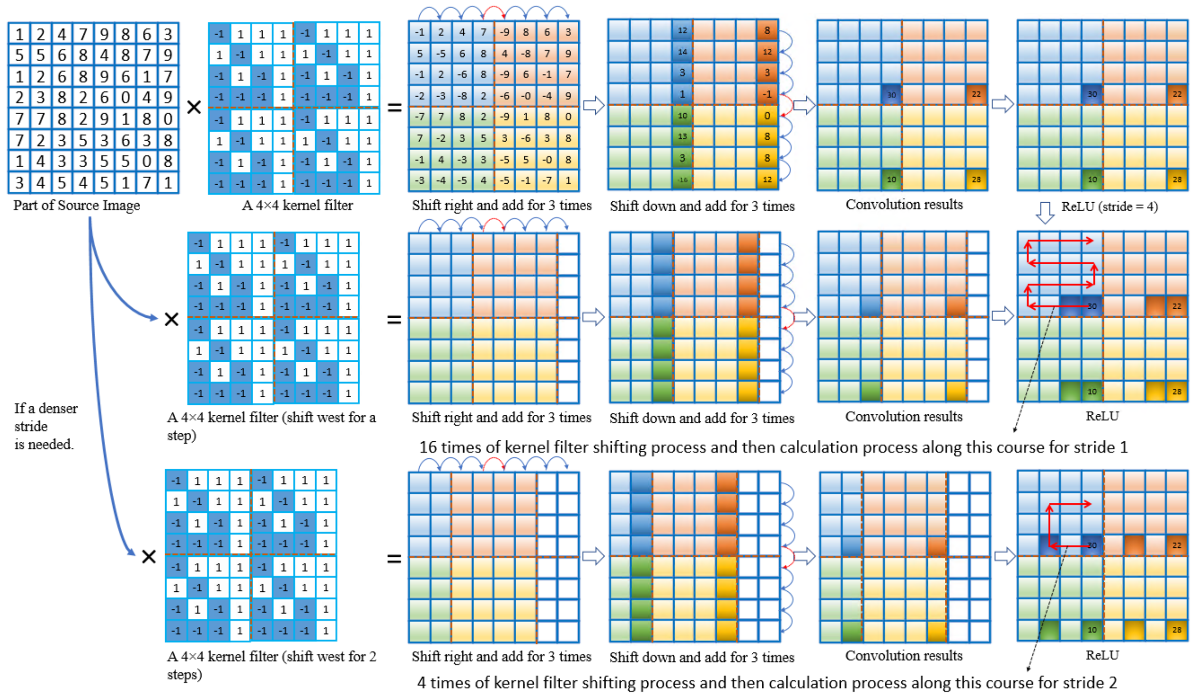

Part of Source Image
A 4×4 kernel filter
Shift right and add for 3 times
Shift down and add for 3 times
Convolution results
ReLU (stride = 4)
A 4×4 kernel filter (shift west for a step)
If a denser stride is needed.
ReLU
16 times of kernel filter shifting process and then calculation process along this course for stride 1
A 4×4 kernel filter (shift west for 2 steps)
4 times of kernel filter shifting process and then calculation process along this course for stride 2

Figure 3.5: The parallel implementation of image convolution process. Only useful information is stored at the right bottom corner in every 4×4 block. The final result in this example can be regarded as a CNN with a stride = 4. Stride can also be set to 1 or 2 according to the requirements of different applications considering efficiency and accuracy.

## 3.4.3 Activation function and Max-pooling layer

Activation function is used in neural network for non-linearity purpose. We make use of the rectified linear unit (ReLU) as it is both a common choice of activation function and can be efficiently performed in parallel across the SCAMP-5d array, using a short sequence of native operations. Max-pooling can similarly be implemented in an efficient parallel manner on the PPA array, using simple shift and addition operations. Specifically 2×2 is achieved by comparing each PE to is north neighbour in parallel, overwriting each PEs data with the larger of the two values. This process is then repeated for each east

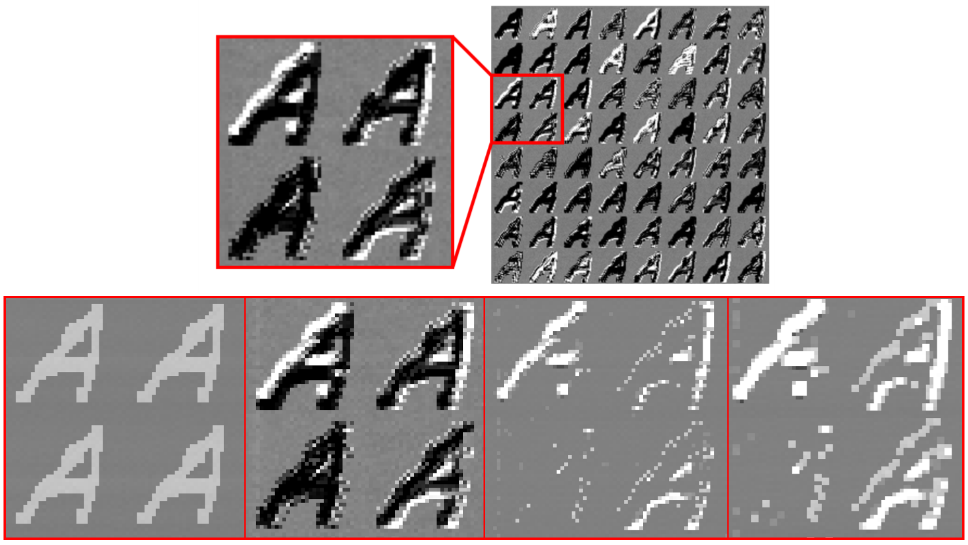


Figure 3.6: Top: 64 feature maps generated in parallel by the convolutional layer on PPA. Bottom (left to right): input images, images after convolution, images after activation function ReLU, images after max pooling.

neighbour, resulting in every PE containing the greatest value in its local 2×2 block.

**Algorithm 4:** Parallel 2×2 max-pooling.

```
Result: B
1 initialisation;
2 D = Move B to the north for one pixel
3 E = D - B
4 WHERE (E > 0)
5     B = D
6 D = Move B to the east for one pixel
7 E = D - B
8 WHERE (E > 0)
9     B = D
```

### 3.4.4 Parallel Fully-Connected Layer

The first step in performing a fully-connected layer is multiplication between max-pooled image data and the fully-connected weights as shown in Figure 3.7. The image on the right visualises the binary weights of the fully-connected layer, encoded in 1-bit digital registers. The key to this part lies in the layout of the fully-connected weights and max-pooled image. In this schematic diagram , the fully-connected weights for 4 labels

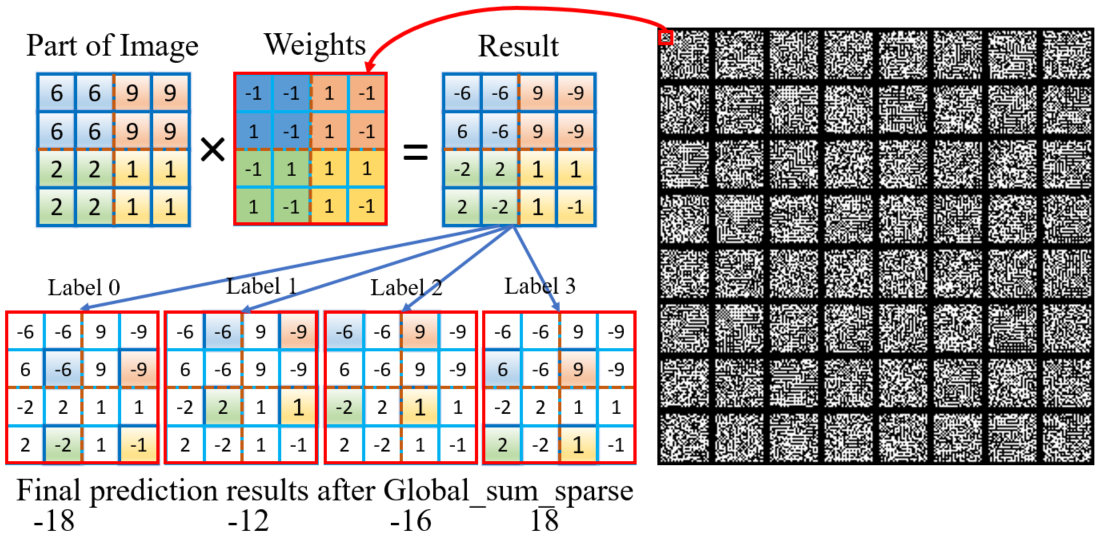


Figure 3.7: The parallel implementation of fully-connected layer.

are stored in the 2×2 blocks. After multiplication, pixels that contain information for each label are spread in a checkered pattern. The native *global_sum_sparse* function can return the approximated summation of values from a given selection of analogue registers. This can then be used to get the approximated sum of pixels associated with each label. The biggest value out of these global summations gives the final prediction of the neural network.

## 3.5 SCAMP-5 Inference, Experiments, and Evaluation

This section demonstrates four experiments[1]: plankton classification, real-time hand-gesture recognition, rock-paper-scissors and digit recognition. Each is demonstrated using a different CNN network running upon SCAMP-5, using either 64 4×4 or 16 4×4 kernel filters in the convolutional layer.

### 3.5.1 Plankton classification

Plankton organisms are at the bottom of the food chain in the marine ecosystem, real-time monitoring of which can be used to determine ocean health levels [170]. Due to the

[1]Experimental video: https://youtu.be/3Qh4ujmsh7E

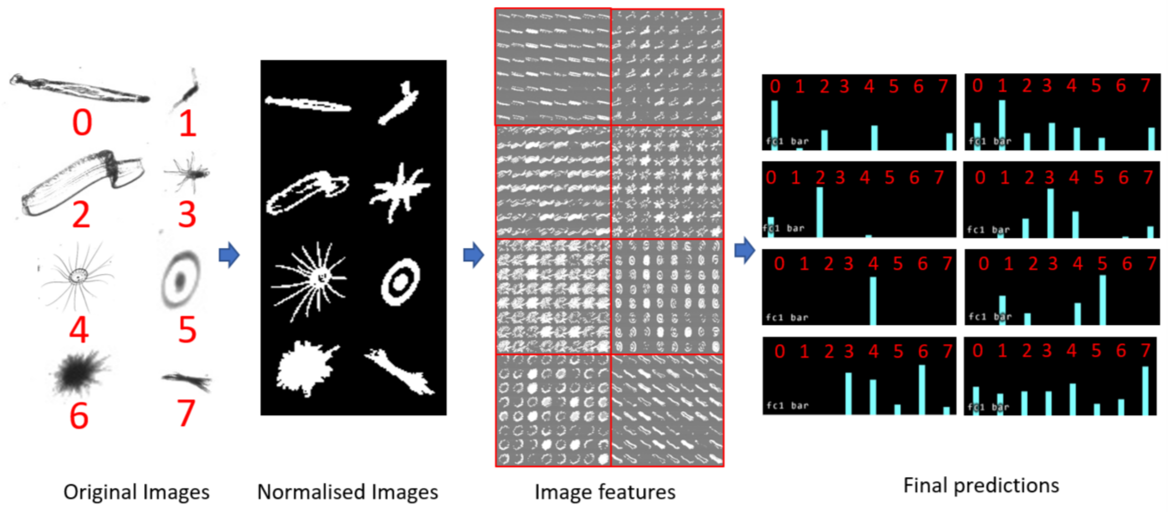


Figure 3.8: CNN inference performing plankton classification on SCAMP-5d. Plankton images are normalised in size and centred before being input into the PPA array as shown in the top row for each class. The second row shows the max-pooled data fed into the following fully-connected layer. Rows three and four show the final predictions for each class and an example image from the correct class.

capacity of the proposed neural network, we select 8 of the most numerous plankton species (0:chaetognaths, 1:coppods, 2:echinoderm, 3:hydromedusae, 4:pelagictunicate, 5:protists, 6:siphonophores and 7:trichode-smium) from an imbalanced scale plankton database considering the number of samples for each species[2], to show the performance of the proposed CNN.

| class | 0.chaetognaths | 1.coppods | 2.echinoderm | 3.hydromedusae | 4.pelagictunicate | 5.protists | 6.siphonophores | 7.trichodesmium |
|---|---|---|---|---|---|---|---|---|
| 0.chaetognaths | **188** | 0 | 1 | 2 | 1 | 0 | 8 | 0 |
| 1.coppods | 3 | **176** | 1 | 0 | 14 | 2 | 4 | 0 |
| 2.echinoderm | 0 | 3 | **182** | 0 | 1 | 1 | 4 | 0 |
| 3.hydromedusae | 1 | 3 | 5 | **181** | 0 | 3 | 7 | 0 |
| 4.pelagictunicate | 0 | 26 | 2 | 1 | **138** | 10 | 23 | 0 |
| 5.protists | 0 | 0 | 1 | 1 | 6 | **183** | 8 | 1 |
| 6.siphonophores | 52 | 12 | 9 | 8 | 24 | 9 | **85** | 1 |
| 7.trichodesmium | 0 | 0 | 17 | 1 | 0 | 20 | 2 | **160** |

Table 3.1: Confusion matrix for plankton classification with 200 samples for each label.

As shown in the Figure 3.8, we utilise 64 4×4 kernel filters, acting upon 32×32 input images with 2×2 max-pooling. After training with binary weight neural network on a computer, the validation accuracy is 83.6% and 80.5% on the PPA. The reason for the accuracy gap lies in the inevitable computation error on analogue registers[171] and approximated analogue summation used in the fully-connected layer. Moreover, Table

[2]Dataset available at https://www.kaggle.com/c/datasciencebowl

1 visualises the performance of the proposed CNN in SCAMP-5 on 1600 samples. The accuracy for siphonophores and pelagictunicate is lower due to their visual similarity with chaetognaths and coppods respectively, which, as a whole, is in line with the bar chart shape in Figure 3.8.

### 3.5.2 Real-time hand gesture recognition

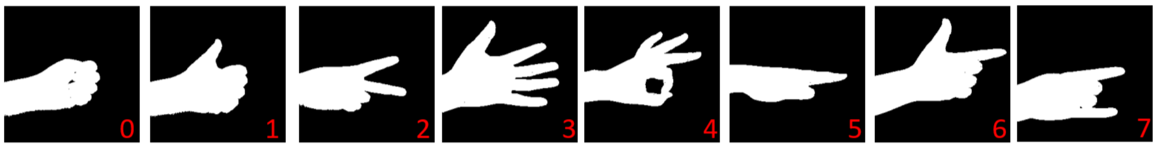


Figure 3.9: Samples of eight common hand gestures for classification with PPA device.

Hand gesture recognition is increasingly used in human-computer interaction, human-robotics interaction and computer games[172]. This section demonstrates real-time hand gesture recognition as another potential application of the proposed CNN framework. The experiment demonstrates real-time recognition of 8 types of hand gesture (Figure 3.9) with image capturing, pre-processing and CNN inference performed on the PPA in a parallel manner.

#### 3.5.2.1 Data collection and Training

We created a hand gestures dataset by capturing commonly used 8 types of hand gestures[3]. Each hand gesture class in the dataset is collected by capturing a dynamic left hand moving randomly within the view-field of the SCAMP-5. More than 1000 images are captured for each class in this way. The CNN used for classification consists of a single 4×4 kernel convolution layer using 16 filters with an input image size of 64×64, followed by a 4×4 max-pooling layer and two fully-connected layers. The choice of two fully connected layers was taken to boost accuracy, with the first performed upon the PPA array and second on the ARM controller. There are 32 intermediate neurons in the first fully-connected layer and 8 in the second. The training with the binary CNN shows the validation result has an accuracy of 98.7% .

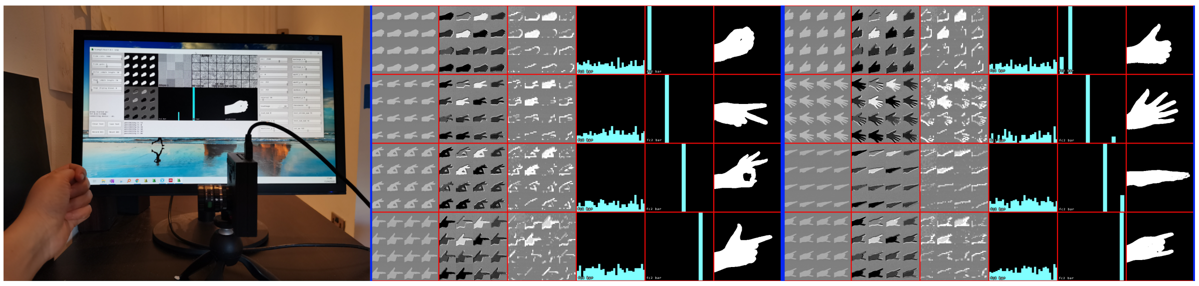


Figure 3.10: Examples of high-speed hand gesture classification by CNN inference on SCAMP-5d. From left to right for each column: (1) Experiment set up showing SCAMP-5d capturing hand gestures while the monitor in the background displays results from the CNN inference being performed on-board. (2) Captured images pre-processed and fed into the CNN, (3) Convolutional layer results, (4) Feature maps after activation and max-pooling, (5) Outputs of the first fully-connected layer and the height of each bar represents value for each neuron, (6) Prediction of the CNN, (7) Visualisation of predicted class.

| Component | Plankton | Hand Gesture | Roshambo | 0 or 1 |
|---|---|---|---|---|
| Image capturing and thresholding ($\mu s$) | - | 6 | 6 | - |
| Character duplication ($\mu s$) | 28 | 28 | 28 | 28 |
| Image convolution($\mu s$) | 165 | 165 | 52 | 12 |
| Activation function ($\mu s$) | 5 | 5 | 5 | 5 |
| Max pooling ($\mu s$) | 4 | 36 | 12 | - |
| First fully-connected layer ($\mu s$) | 47 | 213 | 18 | 12 |
| Second fully-connected layer ($\mu s$) | - | 24 | - | - |
| Total running time ($\mu s$) | 249 | 478 | 121 | 57 |
| Inference speed (fps) | **4,016** | **2,092** | **8,264** | **17,543** |
| Accuracy (Computer/SCAMP-5d) | 83.6%/80.5% | 98.7%/- | 97.73%/- | 99.7%/99.1% |
| Number of binary weights | 100,608 | 921,664 | 43,264 | 29,056 |

Table 3.2: Computation time, performance and weights for different neural networks. Notice that all the live demos are demonstrated with a fixed distance between the SCAMP-5d and the hand.

#### 3.5.2.2 SCAMP-5d Inference and Evaluation

Inference evaluation is performed by a hand randomly changing poses in front of a SCAMP-5d. Figure 3.10 illustrates the prediction results of the proposed neural network. The frame rate of the CNN inference for hand gesture recognition reaches 2092 fps (478 $\mu s$) (Table 5.2).

[3]Dataset available at https://github.com/yananliusdu/scamp/tree/master

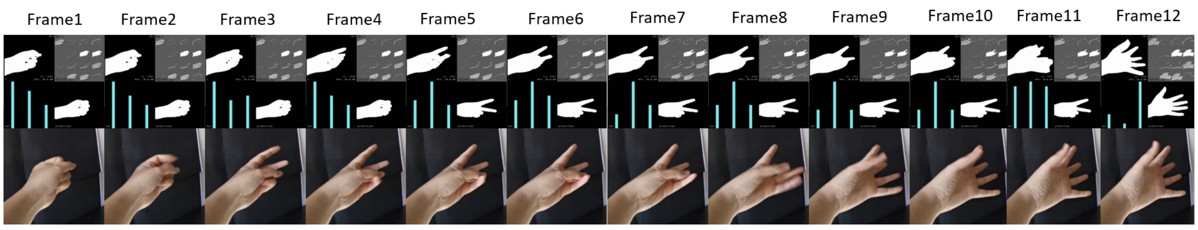


Figure 3.11: Rock-paper-scissors recognition inference process. The image at the bottom is the real hand gesture. Image on the top left is the input for the CNN and the prediction results can be seen at the bottom left for each 4×4 block at the top.

### 3.5.3 High-speed CNN inference on the PPA

To show the high-speed performance of the parallel embedded CNN on SCAMP-5, we implemented a rock-paper-scissors recognition and digit 0/1 recognition with stride = 2 and 4 respectively.

**Rock-Paper-Scissors recognition:**

For this application with 3 labels, a stride = 2 (Figure 3.5) with a single convolutional layer and a fully-connected layer is utilised to achieve a trade-off between the efficiency and robustness. We train a binary neural network with 16 kernel filters on SCAMP-collected hand gesture dataset and get an accuracy of 97.73% (Table 5.2). Figure 3.11 shows the inference process for 12 frames sampled from a 0.3 second period which includes all the time of intermediate result transmission and displaying on the SCAMP-5 host interface for visualisation purpose. Our network can operate with latency of 121 microseconds (from image acquisition to classification result available in the micro-controller), and the frame rate of over 8,200 fps.

**0/1 recognition:**

We trained another network to classify the digits 0 and 1 from the MNIST [173] dataset, to explore how fast CNN inference speed could be pushed for simple tasks. This network uses a single convolutional layer (of stride = 4) followed directly by a fully-connected layer. This approach requires only 12 $\mu s$ for convolutional layer and fully connected layer respectively, achieving a total inference time of only 57 $\mu s$ (Table 2) equivalent to 17,543 fps, and an accuracy of 99.1%.

## 3.6 Direct Servo Control with the PPA

Sensors are usually designed to collect information from the environment and then transmit this information to processors. This section, however, takes advantage of the SCAMP PPA not only to sense and digest input information, but also to send execution instructions driving the servo motors in order to construct a on-sensor 'sense-planning-action' scheme. With the above-mentioned on-sensor neural network, this section develops a direct visual sensory-motor control system using high-speed CNN inference via a SCAMP-5 Pixel Processor Array (PPA). The author demonstrates how PPAs are able to efficiently bridge the gap between perception and action. A binary Convolutional Neural Network (CNN) is used for a classic rock, paper, scissors classification problem at over 8000 FPS during inference. Control instructions are directly sent to a servo motor from the PPA according to the CNN's classification result without any other intermediate or external hardware.

### 3.6.1 Introduction

Real-time image capture, processing, and decision making with low-power consumption are essential for next-generation smart sensors. If a complete neural network inference can be carried out on-sensor, it allows the sensor output to be reduced from entire images to only a small amount of meaningful extracted information used to determine actions. This considerably reduces the communication and energy costs with auxiliary devices. In-sensor visual computing is an area of growing interest [15]. The SCAMP vision system is such an in-sensor visual computing device that performs image processing while sensing and without recording images and while using low power consumption [10, 39]. This is in contrast to a conventional vision system and other non on-sensor computing cameras, where the understanding of the visual inputs happens only after the visual data is transmitted to, and processed by, CPUs and or GPUs, resulting in extra latency and power consumption. This section illustrates how the SCAMP vision system can operate as an 'edge AI' sensor, enabling on-sensor CNN inference and direct action execution with a servo motor, as shown in Figure 3.12.

### 3.6.2 Binary CNN on the PPA

During recent years, there have been many types of CNN binarization methods proposed [174] to minimise CNN model size, improve inference efficiency, and reduce power cost.

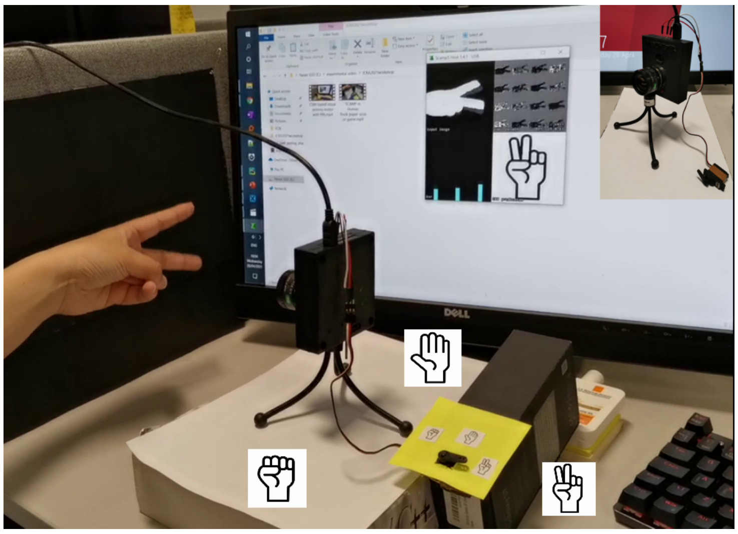


Figure 3.12: A servo motor is directly driven by the SCAMP vision system without additional intermediate hardware. With the machine vision computing results from the SCAMP vision system, the servo can be instructed accordingly. An Align DS425M digital servo is used in this work.

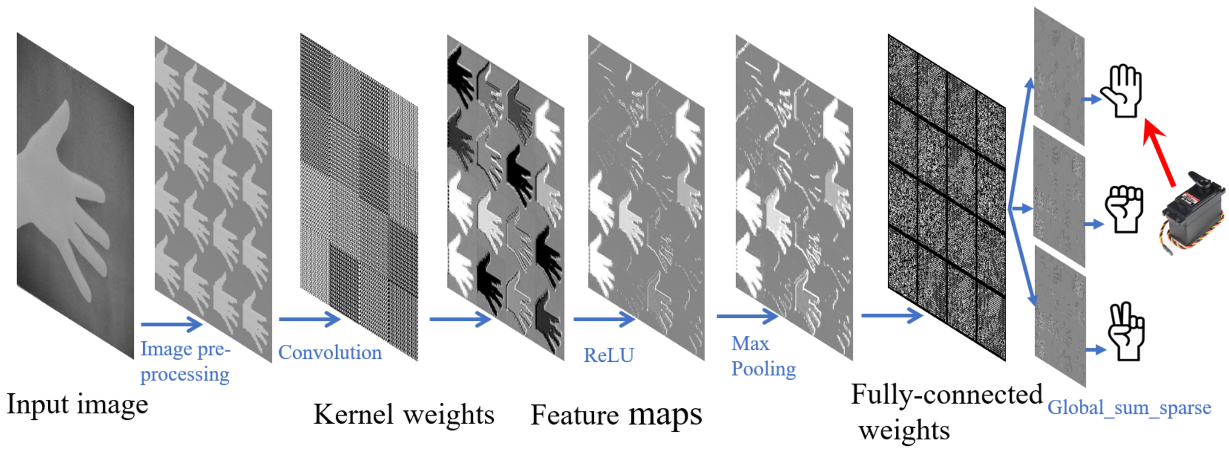


Figure 3.13: Binary CNN inference on the PPA for hand gesture recognition with direct servo control. Image convolution, ReLU activation function, max pooling and fully connected are performed in parallel on the sensing plane.

However, embedded hardware usually has its own dedicated hardware design which is not universally applicable to all forms of binary neural network. To deal with this problem for PPA, this work trains and deploys a binary CNN (with kernel weights and fully connected weights of -1, 1) onto PPA to fully take advantage of its in-sensor parallel computing feature. Binary weights for both convolutional and fully connected layers are stored in the digital registers (DREG) of the SCAMP processor array. The processing

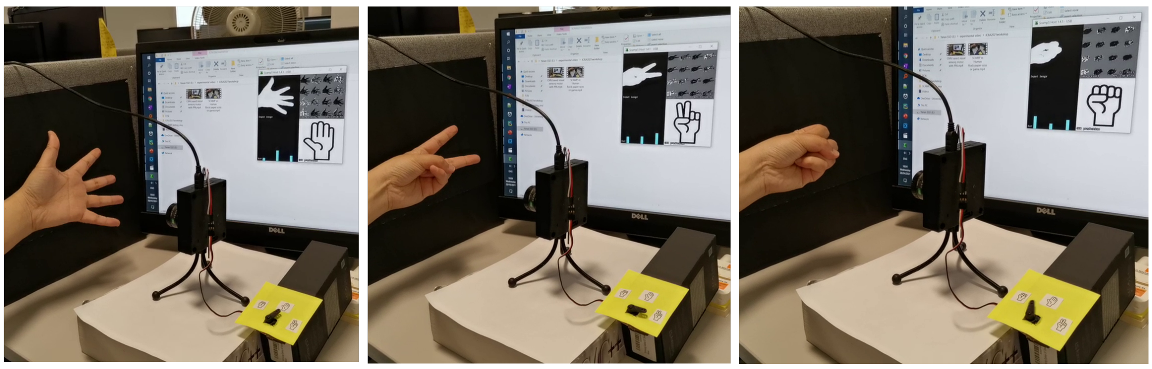


Figure 3.14: CNN inference for hand gesture recognition with a servo as the result indicator.

scheme is illustrated in detail in Figure 3.13. Firstly, the analogue PIX registers acquire light signals, forming the image to be passed through the network. Then the image is binarised, resized and replicated into a dimension of $16 \times 64 \times 64$ using on-sensor processing. Different convolution operations are then carried out simultaneously on these 16 replicated images. Then the activation function ReLU sets negative values to zero. Followed by the $2 \times 2$ maxpooling, a maximum value is selected and replicated within each $2 \times 2$ block. Lastly, the fully connected layer is implemented by activating each label's AREG and then using the SCAMP built-in global summation function to obtain the approximated value for each label. The index for the biggest summation represents the final CNN inference result [65].

### 3.6.3 Experiments

As for the experiment, the SCAMP-5 vision system and a digital servo motor is utilised. Figure 3.12 and 3.14 shows the test-bed for the sensory-motor system. The USB link from the computer is used for power supply and transmission of visualisation data. We extended our previous work [65, 175] by adding the implementation of the direct servo control using the binary CNN inference results. This work is based on the SCAMP-collected hand gesture dataset which consist of gestures for 'rock', 'paper', and 'scissors'. The servo motor is controlled through pulse-width modulation on the ARM Cortex M0 processor embedded in the SCAMP system, where the computing results are shared from the PPA. We perform the experiments on three types of hand gesture recognition. This basic setup is used here as a test-bed for bridging perception to action. After the implementation of CNN on the SCAMP, the experimental results (details can be seen from the video results) offer recognition accuracy of >97% with a latency of 121$\mu$s and maximum theoretical throughput of 8264 FPS [65]. Note the data transmission to a computer for visualisation slows down the overall FPS, which can be noticed from the experimental video. In addition, there are additional latencies, including the servo reaction and movement time until indicating the correct predictions with the servo motor. The control frequency determines the servo reaction time. Our work uses a digital servo motor with a control frequency of 333 Hz leading to approximately a 3 *ms* latency.

The experimental video can be seen at *https://youtu.be/gHcuv275Qrk*

with a ×20 slow motion available from *https://youtu.be/SAMsIqqCZ7I*

## 3.7 Discussion

The proposed new implementation of convolutions allows more flexibility (different strides and different max-pooling setup) to modify a CNN for different tasks and achieves higher speeds 2,000-17,000 fps. Compared to works [63, 113, 114] which only test on MNIST, we expand to Plankton and 2 live hand gesture tasks. [63] uses ternary-weighted CNNs and achieves 94.2% at 210 fps. [113] claimed it reaches 2260 fps and quoted an accuracy of 96.9% on MNIST, but only uses 3 convolutional filters which may be insufficient to generalise to other tasks. Moreover, its frame rate drops to around 1000 fps with 7 convolutional filters indicating the nature of parallelism on the PPA is not fully exploited. [113] implemented both max-pooling and fully-connected layers in Micro-controller and the maximum inference reaches 3000 fps with a sacrificed accuracy of 90.2%.

The bottleneck that limits further performance improvement on SCAMP-5 in terms of accuracy and speed is due to the insufficient engineering resources available to academic research. If the PPA is built with state-of-the-art technology (current PPA device is manufactured with 180 *nm* CMOS silicon technology [10]), these limitations will be greatly mitigated. Finer silicon process implementation will provide more digital storage per pixel and an expanded ALU, while silicon stacking technology allows extra advantages of analogue pixel computing to still be exploited (e.g. low power, global sum, blur, etc).

## 3.8 Conclusion

In this work the author demonstrated performing CNN inference upon a PPA sensor-processor device across various tasks. Our implementation exploits the parallel computation of the entire PPA array, compared to various previous work which only utilised a small area. As a result our CNN inference is shown to be significantly faster than these works. Further our proposed convolution approach allows convolutions of stride 1,2 and 4 enabling extremely high inference speeds over 17500Hz on certain tasks to which stride 4 is applicable. The range of tasks demonstrated illustrate the potential such PPA devices may hold for future embedded applications. Though the current limitations of PPA hardware restrict us to smaller networks, it is reasonable to assume that future devices will see a significant increases in PE memory, power efficiency, and processing speed. The work presented here could quickly be adapted to take advantage of such improvement and thus can be used as a stepping stone towards more complex computational vision applications.

In addition, a visual sensory-motor scheme based on the PPA is presented in this work, where a servo is directly controlled by the focal-plane CNN inference without additional computing units. We believe that closing the gap between perception and action, through in-sensor computing strategies, is essential for agile and efficient robotic systems.

CHAPTER 4

# ON-SENSOR BINARISED CONVOLUTIONAL NEURAL NETWORK

The preceding chapter proposes and implements a binary neural network on the SCAMP PPA for a variety of classification applications. The trained binary neural network, on the other hand, just binarises the weights. There is still a large number of activations that require at least 8 bits of storage space for each neuron. Additionally, AREG is unsuitable for storing activations momentarily, as the analogue current within the integrated circuit quickly becomes saturated, introducing noise. Keeping these constraints in mind, a fully binarised neural network may hold the answer to overcoming these obstacles. As a result, this chapter proposes a neural network with both binary weights and binary activations.

## 4.1 Background

Neural network quantisation is a critical technique for condensing massive floating-point networks into much smaller networks that may be implemented on resource-constrained mobile and portable devices. Binary neural networks are one technique for compressing a neural network effectively for deployment on embedded systems. There is a great deal of research being conducted on neural network binarization at the moment. To accomplish high-performance CNN inference on embedded devices, extensive research on network compression, hardware accelerators, and novel visual sensors has been conducted. This thesis trains binary neural networks using an off-sensor technique in preparation for network deployment on the PPA device for various applications, including classification, localisation, and segmentation.

Many types of binarised CNN training methods have been proposed in recent years, aiming to provide an efficient and compressed neural network solution for embedded edge devices with limited computation power and memory. However, it can be challenging to find an optimal method to deploy such neural networks onto certain devices because of their non-standard architecture, such as the case for Pixel Processor Arrays (PPAs) sensors. This work attempts to enable a network solution for the PPA by training purely binarised CNNs and deploying them on the PPA's focal plane. We demonstrate Convolutional Neural Network (CNN) and Fully Convolutional Network (FCN) with tasks of classification, localisation, and coarse segmentation. Specifically, this work trains and implements networks with batchnorm and adaptive threshold for binary activations. Then, we convert batchnorm and binary activations into a bias matrix which can be parallelly implemented by an add/sub operation. Considering the limited computing and storage hardware resources of PPAs, we propose a novel dynamic model swapping scheme to decompose applications that are beyond the capacity of PPAs into sub-tasks that the tree networks can solve. Thus, our approach demonstrates various tasks on a highly resource-limited PPA sensor chip without using external computing units.

## 4.2 Introduction

Sensing, storage, and processing integration wholly on a single chip based on analogue signals can be a potential solution for the next-generation intelligent embedded sensors [41], where artificial neural networks can be performed in real-time. With this all-in-sensor scheme, a sensor can efficiently perceive the environment and generate useful results with the raw analogue signals using low-power consumption without producing redundant data to post-processing units. Pixel Processor Array (PPA) (Figure 1.2) [10] is such an emerging programmable massively cellular vision the sensor that combines imaging, information storage, and computation in-sensor. Considering the decentralised Processor Elements (PEs, Figure 1.2) of PPA, the neural networks need to be transformed to fully take advantage of its in-sensor parallel processing performance for more complicated tasks. However, full-precision neural networks usually require huge ROM and RAM space for weights and temporal results storage along with heavy floating-point computation, which is adaptive with powerful CPU/GPUs but not suited for on-sensor computing devices with only stringent storage and computation hardware resources.

With this in mind, this paper designs and trains new purely binarised CNNs adaptive to PPA's hardware architecture and proposes novel methods to deploy these proposed CNNs on the PPA across image classification, object localisation and coarse segmentation tasks.

However, here comes another problem: binarised CNN with binary weights and activations are difficult to train and usually suffer from significant performance drop [176] compared to full-precision CNN training. Batch norm [177] is an essential technique in training binary neural networks to avoid gradient explosion and help the neural network to converge [176]. As a result, we are motivated to introduce batch norm into the purely binarised CNNs to improve both the training efficiency and performance in the embedded vision systems with scarce storage and computing resources by adding more layers. In this paper, to fully take advantage of PPA's parallel computation ability on both analogue registers (AREG) and digital registers (DREG) (Figure 1.2, [39]), we train a purely binary convolutional network containing binarised weights and activations with batch norm off the sensor and deploy it on the sensor for multiple tasks. We show that the implementation of the batch norm, activation function, and adaptive activation threshold for binarised activations is mathematically equivalent to adding a bias matrix $\boldsymbol{B}$ which significantly simplifies the inference process on the sensor. In addition, the use of binary activations alleviates accumulative errors introduced by analogue-signal based computing when performing image convolution and allows an accurate bit-counting method for the fully-connected layer, hence enabling deeper CNN on the sensor. With this binarised CNN, the inference calculation process can be implemented with only add/sub and shifting operations for convolution and binary bit counting for the linear layers. This scheme benefits explicitly the SCAMP vision system because the calculation errors are easily accumulated when using analogue current signals repetitively [171] for storage and calculation especially for activations. It is also worth noticing that AREG calculation is based on analogue current signals that are subject to saturate after repetitive operations resulting in information loss. In this work, binary weights and neuron activations can help to decrease the accumulative errors and avoid AREG saturation issues. Furthermore, this presented neural network binarization and on-sensor implementation architecture are generalised to CNN with linear layers and Fully Convolutional Networks (FCN) for various tasks.

In the experiments, the architecture of the CNN tree with dynamic model swapping is proposed to enable embedded devices with limited hardware resources to deal with more complicated tasks by switching different network models. Specifically, we send input images through a sequence of networks. At each stage, the output of the last network is used to determine which network is next, effectively allowing a more complex task to be performed by a series of simple networks. We apply this scheme to classification, object localisation and segmentation tasks. We implement this CNN tree architecture on the sensor and demonstrate it with real-time 37 types of English letters classification and objects' localisation and different coarse segmentation. The main contents of this work can be summarised as follows: (1) We train purely binarised CNNs (binary weights and activations) and implement them on the PPA for the first time. This approach of binary activation alleviates the accumulation of analogue computing errors and value saturation after each layer, thus enabling deeper neural networks on the sensor. Furthermore, with the binary neuron activations as inputs, the linear layers can be implemented by simply counting the bit number accurately [65]. Unlike earlier work, we apply batch norm during training and utilise this to learn bias parameters to be applied during inference on the PPA device. (2) We propose a CNN tree architecture where multiple neural networks can be composed for more sophisticated inference tasks by dynamically uploading neural network models onto the sensor.

### 4.2.1 CNN with Binary Weights and Activations

BinaryConnect [89] trains neural network with binary weights during forward propagations. However, BinaryConnect is not a fully binary neural network with floating-point neuron activations. Both the CNN and FCN presented in this work are based on the binarised CNN [178] with both binary weights and neuron activations. Such binary values can be stored in 1-bit DREG and processed with bit-wise operations upon the PE array. binarised CNN reduces the intermediate memory storage required for neuron activations and replaces most arithmetic operations with bit-wise operations. These qualities make such fully binarised neural networks are highly suitable for PPAs. During training we employ a simple strategy to binarise the weights and activations. All the weights are efficiently binarised in a deterministic manner Equation 4.1.

$$w_b = Sign(w_r) = \begin{cases} +1 & w_r > 0, \\ -1 & otherwise \end{cases} \tag{4.1}$$

$$a_b = Sign(a_r - \alpha) = \begin{cases} +1 & a_r > \alpha, \\ -1 & otherwise \end{cases} \tag{4.2}$$

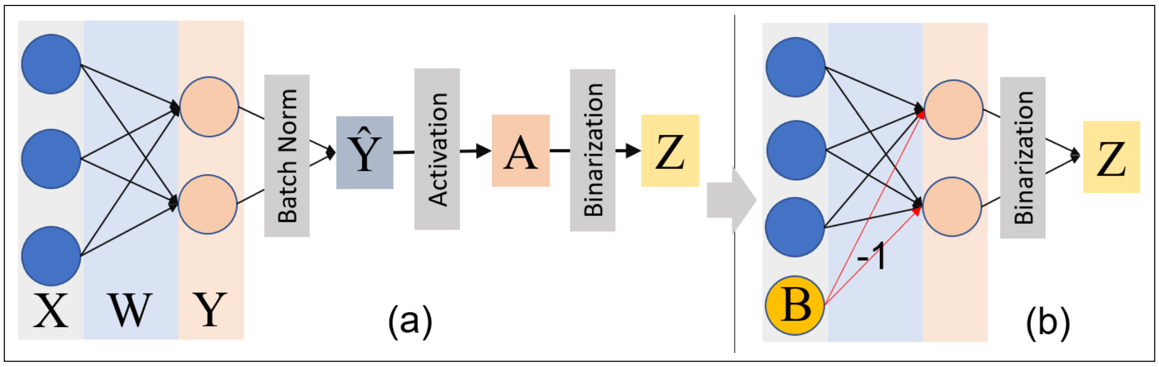


Figure 4.1: (a) The binarised CNN forward propagation with batch norm and adaptive activation function. (b) The simplified inference process on the PPA device by transforming batch norm and activation function into a 'bias' $\boldsymbol{B}$ to be subtracted from $\boldsymbol{Y}$. The inference process is significantly simplified, with only addition/subtraction and sign operations required.

where $w_r$ is floating-point weights and $w_b$ is the binarised weights. In terms of activations, we train channel-wise adaptive thresholds $\alpha$ to binarise the activations to obtain more informative binary feature maps, inspired by work [179]. Additional coefficients, introduced by channel-wise thresholds, have a low impact on the implementation efficiency. In Equation 4.2, $\alpha$ is the trainable thresholds for binarization of each channel, $a_r$ is the real-valued activations and $a_b$ is the binarised activations. During the training process, the gradients are calculated with the floating-point weights using standard backpropagation and stochastic gradient descent. The weights and activations are only binarised during the forward pass. In our work, the binarised CNN is trained on a PC machine, and the CNN inference process is implemented on the SCAMP-5d vision system.

The training process for batch norm parameters can be seen from Algorithm 5 [177].

In this batch norm training algorithm, $\epsilon$ is used to avoid a zero denominator and the main scaling and shifting parameters $\gamma$ and $\beta$ for batch norm are learned during the training process. Then the batch norm can be applied to manipulate activations [177]. In Figure 4.1, for a single layer of binarised CNN during the forward propagation process:

$$Y = \sum_{i=0}^{m} w_i x_i, \hat{Y} = \gamma \frac{Y - \mu}{\sqrt{\sigma^2 + \epsilon}} + \beta = \frac{\gamma}{\sqrt{\sigma^2 + \epsilon}} (Y - (\mu - \frac{\sqrt{\sigma^2 + \epsilon}}{\gamma} \beta)) \tag{4.3}$$

Considering activation function $tanh$ does not change the sign of inputs, we have:

**Algorithm 5:** Batch norm parameter training

```
1 Input: Mini-batch $B = \{x_1, x_2, ..., x_m\}$
2 Output: $y_i = BN_{\gamma,\beta}(x_i)$
    1. Calculate the average for each mini-batch: $\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i$.
    2. Calculate the variance for each mini-batch: $\sigma_B^2 = \frac{1}{m}\sum_{i=1}^{m}(X_i - \mu_B)^2$.
    3. Normalise the inputs: $\hat{x}_i = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}}$.
    4. Scale and shift: $B_i = \gamma \hat{x}_i + \beta$.
```

$$Z = sign(A) = sign(tanh(\hat{Y} - \alpha)) = sign(\hat{Y} - \alpha) = sign(Y - (\mu - \frac{\sqrt{\sigma^2 + \epsilon}}{\gamma}\beta) - \alpha) \tag{4.4}$$

$$Z = sign(Y - B) \tag{4.5}$$

$$B = \mu + \alpha - \frac{\sqrt{\sigma^2 + \epsilon}}{\gamma}\beta \tag{4.6}$$

In Equation 4.6, $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2, \mu = \frac{1}{n}\sum_{i=1}^{n} x_i$, where $\beta$, $\gamma$, and $\alpha$ are all trainable parameters that can be obtained directly after training. Thus the 'bias' $\boldsymbol{B}$ can be calculated used these parameters offline, before implementing it on the PPA.

During the inference process, the batch norm and activation reduces to a bias term, as shown in Equation 4.5 $\boldsymbol{B}$ on $\boldsymbol{Y}$. Hence, the inference process on the PPA can be simplified as shown in Figure 4.1b.

## 4.3 Purely binarised CNN Training

The training method of binary neural network has been illustrated in Section 3.3 for the binary neural network on the sensor. This section aims to find a neural network training methods to generate not only binary weights but also binary activations in order to deploy a deeper neural network on the PPA. This purely binarised CNN training method is inspired by work [178] where bit-wise operation can replace most arithmetic operations, which is suitable for the PPA architecture supporting parallel bit operations on DREG.

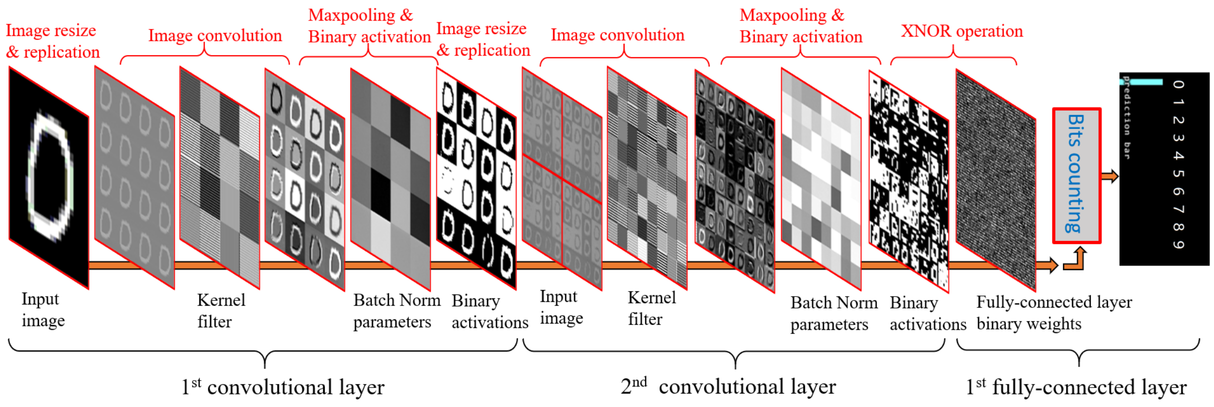


Figure 4.2: CNN inference process with multiple layers on the PPA by integrating image sensing, storage and calculation using both DREG and AREG. Extra fully-connected layers can be extended on the micro-controller for more complicated tasks.

### 4.3.1 Forward propagation

During the training process, forward propagation generates the inference result using the sub-optimal weights to further compare it with the ground truth and obtain the loss. The forward propagation is similar to Section 3.3.2 where binary weights are utilised. The only difference in the binarised network is to binarize the activations after the convolutional layer.

### 4.3.2 Gradient

The gradient is critical for backpropagation and convergent training throughout the neural network training process as the full-precision weights are updated iteratively using the Stochastic Gradient Descent (SGD). The derivative of the binarization/threshold, on the other hand, is 0, resulting in training failure. To address this issue, the straight-through estimator is used to pass on the incoming gradient and treat the derivative of the binarization function as an identity function [180]. The procedure of training binarised neural networks was inspired by the work [178] and can be seen in Algorithm 6. Notably, we $Clip$ the weights during training, keeping them within the range [-1,1], which prevents the weights from becoming overlarge.

**Algorithm 6:** binarised neural network training process.

```
1  Definitions:
2  $C$: cost function,
3  $\lambda$: the learning rate decay ratio,
4  $L$: the number of layers,
5  $\circ$: element-wise multiplication
6  Inputs:
7  a mini-batch inputs and targets $(a_0, a^*)$,
8  previous weights $W$,
9  previous batch normalisation parameters $\theta$,
10 weights initialisation coefficients $\gamma$,
11 previous learning rate $\eta$,
12 Ensure:
13 updated weights $W^{t+1}$,
14 updated batch normalisation parameters $\theta^{t+1}$,
15 updated learning rate $\eta^{t+1}$,
16 1. Forward propagation:
17 for $k = 1$ to $L$ do
18     $W_k^b \leftarrow Binarize(W_k)$
19     $s_k \leftarrow a_{k-1}^b W_k^b$
20     $a_k \leftarrow BatchNorm(s_k, \theta_k)$
21     if $k < L$ then
22         $a_k^b \leftarrow BatchNorm(a_k)$
23 2. Backward propagation:
24 $Compute\ g_{a_L} = \frac{\partial C}{\partial a_L}\ knowing\ a_L\ and\ a^*$
25 for $k = L$ to $1$ do
26     if $k < L$ then
27         $g_{a_k} \leftarrow g_{a_k^b} \circ 1_{|a_k| \leq 1}$
28     $(g_{s_k}, g_{\theta_k}) \leftarrow BackBatchNorm(g_{a_k}, s_k, \theta_k)$
29     $g_{a_{k-1}^b} \leftarrow g_{s_k} W_k^b$
30     $g_{W_k^b} \leftarrow g_{s_k}^T a_{k-1}^b$
31 3. Accumulating the parameters gradients:
32 for $k = 1$ to $L$ do
33     $\theta_k^{t+1} \leftarrow Update(\theta_k, \eta, g_{\theta_k})$
34     $W_k^{t+1} \leftarrow Clip(Update(W_k, \gamma_k \eta, g_{W_k^b}), -1, 1)$
35     $\eta^{t+1} \leftarrow \lambda \eta$
```

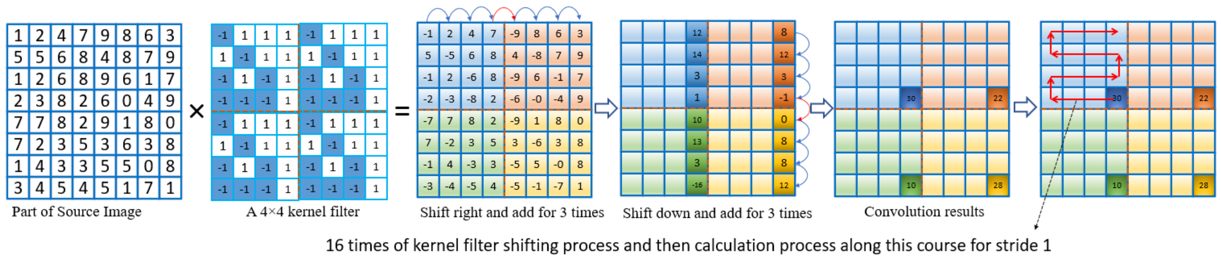

Part of Source Image
A 4×4 kernel filter
Shift right and add for 3 times
Shift down and add for 3 times
Convolution results
16 times of kernel filter shifting process and then calculation process along this course for stride 1

Figure 4.3: Image convolution on PPA with sign inversion, bit-shifting, and addition.

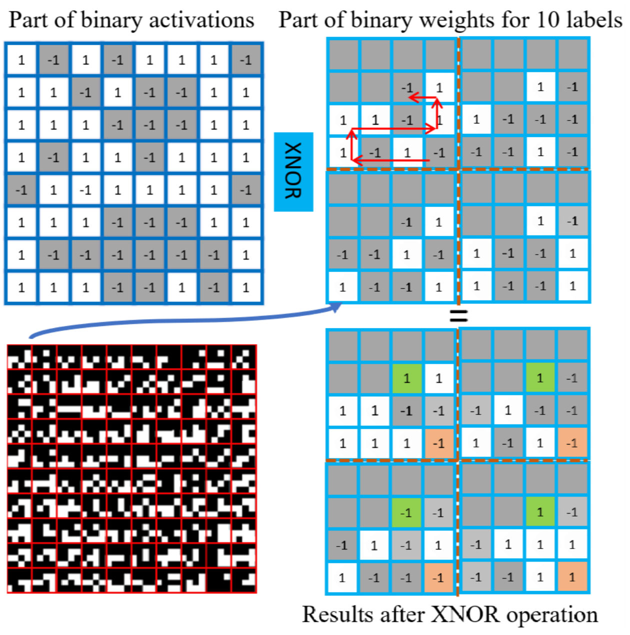

Part of binary activations
Part of binary weights for 10 labels
XNOR
Results after XNOR operation

Figure 4.4: The fully-connected layer on PPA using parallel XNOR operation and bit counting. The binary weights for labels are stored on DREG along a ‘snake’ pattern. For example, the results for label 0 can be obtained by counting the sum of values in orange boxes ($-1-1-1+1=-2$). The results for label 9 from green boxes ($1+1-1+1=2$).

## 4.4 Purely binarised CNN Implementation Method on the PPA

It is critical to carefully design the CNN architecture in order to maximise resource utilisation and parallelism especially for the unique hardware design of PPA. The overall binarised CNN architecture from sensing, storage, and computing can be seen from Figure 4.2. Firstly, as for imaging, the photodetector (PIX) converts light into an analogue signal which can be directly transformed and temporarily stored into AREG. The input image on AREG is then resized and replicated to full fill the whole 256×256

PPA for parallel processing purposes. With the binary convolutional weights stored in DREG, the image convolution can be performed using the replicated image and its associated weights. As illustrated in Section III, the batch norm can be efficiently implemented by subtracting a matrix that is plotted in AREG. The binary activations can be obtained by binarizing the full-precision activations after being subtracted by the bias matrix. Then the binarised activations act as the inputs for the next convolutional layer using a similar process as the first layer. The input for the fully-connected layer is the binarised activations, with binary weights. The final prediction can be generated by performing *XNOR* operation and counting the number of bits for each label. The maximum number of bits indicates the final CNN inference result. The following sections detail the implementation method for each layer.

### 4.4.1 Convolutional Layer

As shown from Figure 5.5, 4.2, and 5.3, the image convolution can be parallelly executed on the PPA by 'multiplications', shifting, and add/sub. Firstly, the register which stores the image information gets its sign inverted or remain unchanged according to the sign of binary weights stored in another DREG (Figure 5.3). Then the convolution operation can be performed by shifting and adding horizontally three times and vertically for another three times. The first image convolution result (Figure 5.3 Convolution results) can be found in the bottom-right cell out of each 4×4 block. Finally, after repeating this process for 16 times (shown in the last plotting of Figure 5.3), an image convolution with stride one can be obtained. Image convolution with different strides can be implemented by different shifts [65]. Figure 5.5 shows the layout of the convolutional kernel filters in detail. Each 4×4 kernel filter are replicated into a 64×64 block (16 blocks in total) to extract corresponding features parallelly. This work implements a CNN with two convolutional layers, the resolution of input images to PPA is 256×256, and after resizing and replication, 16 64×64 images are inputs for the first convolutional layer followed by 2×2 maxpooling. For the second convolutional layer, a group convolution [181] with 16 groups, and 4×4 max-pooling is utilised to simplify the calculation, reduce the memory requirement, and accelerate the network inference process. The input image for the second convolutional layer (shown in Figure 4.2) contains four identical images (128×128). In each of them, there are 16 feature maps. After an image convolution with a group of 16, 64 feature maps are generated simultaneously, acting as inputs for the next CNN layer. With this method, there is no need to shift feature maps and add/sub them into a new one within no matter AREG or DREG for the second convolutional layer, hence calculation

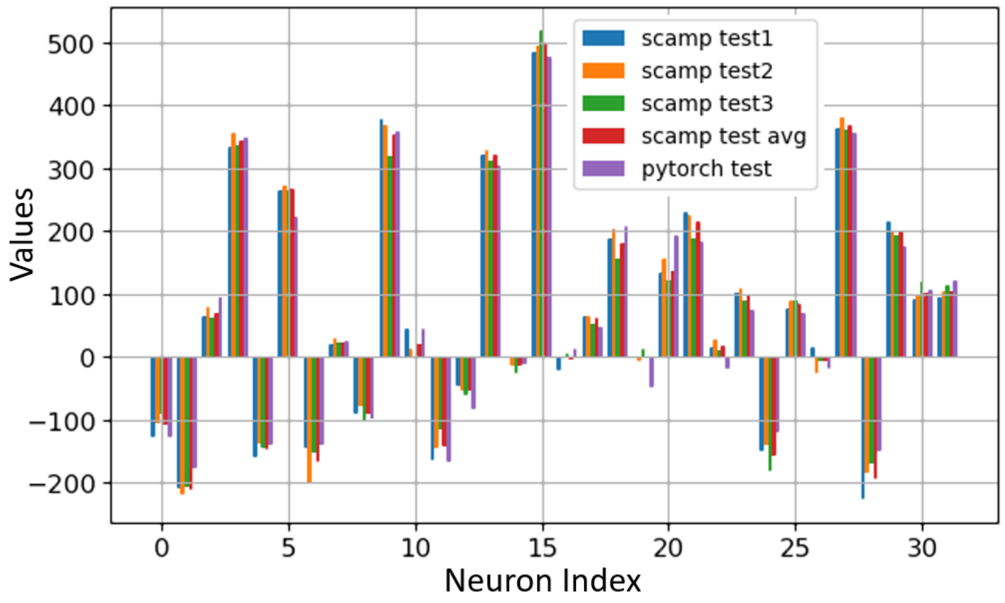


Figure 4.5: CNN neuron values (32 neurons in the first fully-connected layer) comparison between PPA and PyTorch simulation for the first fully-connected layer. A single image is tested three times on SCAMP and neuron values, and their average values are recorded to compare with ground-truth values from PyTorch.

errors based on the analogue signals can be reduced, and the inference process can be accelerated with much less register-based operations.

### 4.4.2 Fully-Connected Layer by Bit Counting

As shown in Figure 4.4, the first step is to multiply input binary feature maps with 1-bit weights, which is achieved by performing *XNOR* operation between the given binary activations (Figure 4.4 top left) and weights (top right), which can be efficiently processed with the parallel bit operation on DREG. Figure 4.4 shows the weights for 10 labels and they are pre-stored in a 'snake' pattern in a 4×4 block on a DREG. Different from earlier work [64, 65, 110] where the fully-connected layer is implemented mainly by using the built-in *scamp5_global_sum* function to estimate the summation of values in AREG, which is not accurate to fully represent the possibility of the CNN outputs. In our work, the final CNN reference possibility can be obtained by activating each label's position in the bottom right (Figure 4.4) and counting the amount of positive and negative bits, which can be more accurate than approximation summation method. The pixel counting accuracy can be found from Figure 4.5, which shows the similarity between a simulation on pyTorch that can be regarded as ground truth and binarised CNN on the PPA. An

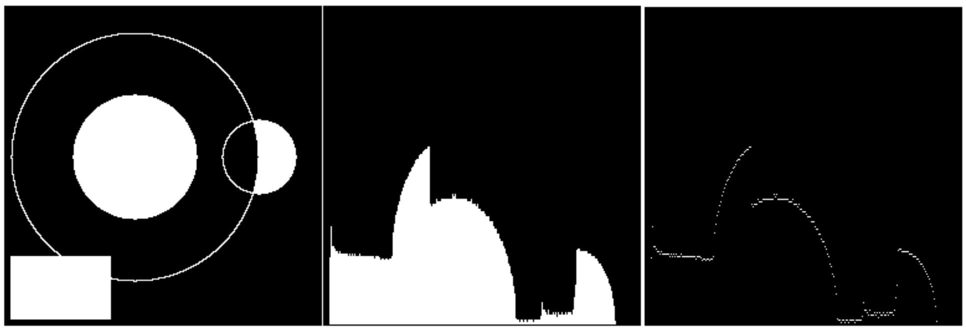


Figure 4.6: The stages of the bit counting method known as ‘sandcastle summation.’ From left to right, an example of DREG content to be summed, the stacked ‘sandcastle’ of set pixels that results, and the pixels along the top of the ‘sandcastle’ whose extracted locations are used to determine the total number of set pixels.

average error of 10-15 is achieved between the groundtruth and values on the PPA for each neuron.

**Bit counting for fully-connected layer:**

When calculating the neuron activations from the fully connected layer, counting the number of set bits in a DREG is necessary. In this work the author make use of the "sandcastle summation" method introduced in [64], and described in detail in [182] [1]. In brief, this method provides a fast way to calculate an exact count of the number of set bits in a DREG. It achieves this by manipulating the DREG’s content via efficient parallel operations into a form where the number of set pixels can be easily determined. Specifically forming a stacked "sandcastle" of set pixels as shown in Figure 4.6 after which, the number of set pixels can be determined by calculating the area of this "sandcastle" stack. This approach is typically two orders of magnitude faster than a naive approach of counting set pixels individually.

### 4.4.3 Activation and Max-pooling

As mentioned above, the activation function *tanh* is simplified in the binarization process—hence no need to implement the activation function in the binarised neural network. Max-pooling is implemented in the same way as that in Section 3.4.3.

[1] https://youtu.be/a2VO3aWHnYc

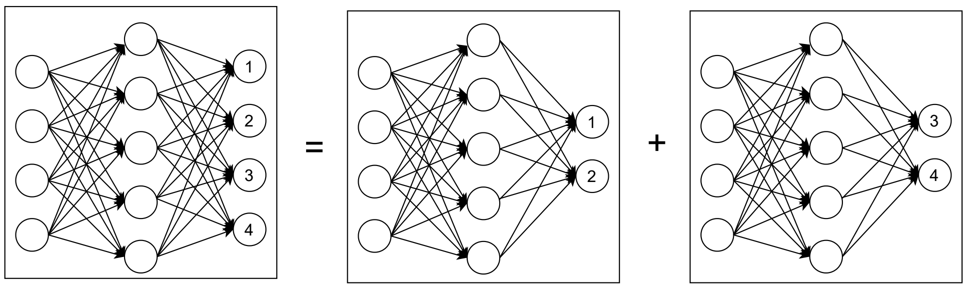


Figure 4.7: A schematic diagram: a ‘bigger’ network is directly decomposed into multiple ‘smaller’ networks.

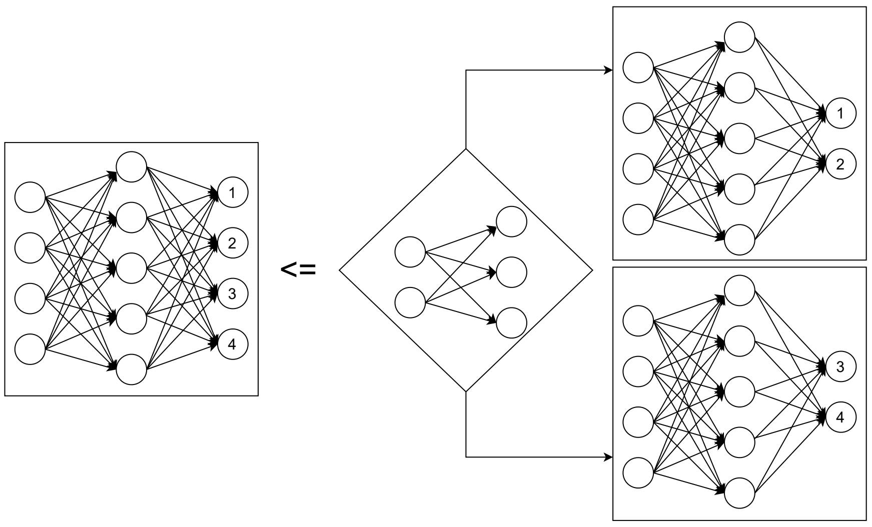


Figure 4.8: A schematic diagram: a ‘bigger’ network is decomposed into multiple ‘smaller’ networks with a selection network to choose which one to run next.

## 4.5 Dynamic Network Model Swapping on the PPA

Considering the limited AREG and DREG resources on the sensor, a dynamic model swapping scheme can be a solution to perform multiple networks in sequence towards a sophisticated application. Specifically, these multiple networks can be trained separately to approximate a comparatively "large" network that is out of the storage and computing capacity of the PPA sensor. The PPA can obtain similar or even better inference results by sequentially running these "smaller" networks.

As shown in Figure 4.7 and 4.8, there are mainly two schemes to decompose a network into smaller ones: the direct decomposition and decomposition with an extra selection network. Figure 4.7 shows the direct decomposition where classes are divided into different groups and sub-networks are executed in sequence to find the inference results.

In terms of the inference results, it is evident that the combination of sub-networks would not be better than the original one. Different from the direct decomposition scheme, Figure 4.8 introduces an extra selection network to decide which neural network to carry out next according to the inference result from the selection network. Each sub-network is trained separately in this scheme, and the overall performance can be better than the original network. In addition, more sub-networks are needed with more labels to classify, for example, alphabet classification. In this case, the PPA needs to run all sub-networks one by one to get the final inference results with the scheme shown in Figure 4.7 while only one sub-network is needed after the selection network with Figure 4.8. Hence, this thesis takes advantage of sub-networks with a selection neural network for many labels' classification by dynamically swapping only two sub-networks. In addition, the neural network models are stored in the FLASH of the SCAMP vision system. Because of the same neural network architecture, the associated weights for models can be uploaded into DREGs for computing after the last inference result.

## 4.6 Experiments

This section demonstrates experiments, classification, localisation, and segmentation, based on the proposed binarised CNN and FCN architecture, respectively. Figure 4.2 shows the design of a binarised CNN architecture as a node of CNN tree for 37 English letter (Figure 4.9) recognition. Figure 5.11 shows the CNN architecture for object 2D localisation and road coarse segmentation with FCN.

### 4.6.1 CNN Tree on EMNIST for English Letter Classification

The EMNIST dataset [183] is a set of handwritten characters extended from MNIST [173] dataset with the same image format. This work uses the merged class which contains 11 lower-case classes, 11 upper-case classes and 15 mixed classes where some of letters are difficult distinguish from upper case to lower case, such as *O, X, C*. Hence, there are 37 types of labels in the merged class in total. Considering the scarcity of hardware computing resources, especially the amount of DREG/AREG available to store weights, temporary activations and perform convolution, this paper proposes a CNN tree architecture consisting of 4 CNNs (Figure 4.10), each of which can be swapped simply by loading associated weights from the flash to the DREG, as all these four networks share the identical architecture. The overall 4 CNN structure shown in Figure 4.10. The three

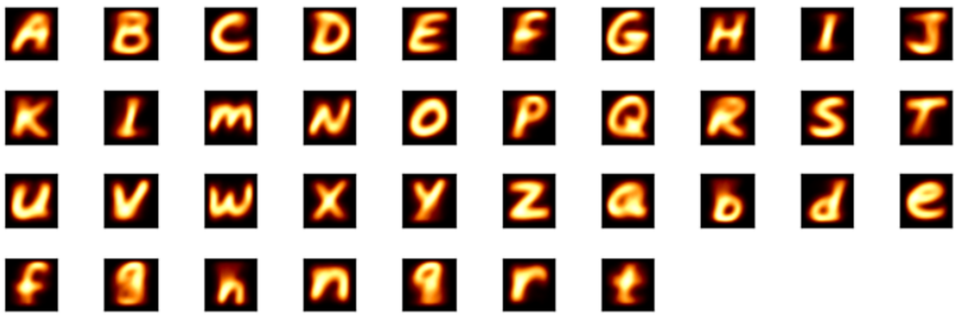


Figure 4.9: The visualisation of the average value for each letter from the EMNIST dataset. As can be seen, several classes share lots of similarity to some degree, such as *I and L*, *f and F*, *h and n*, *g and q*, *Q and a* which makes this classification task challenging.

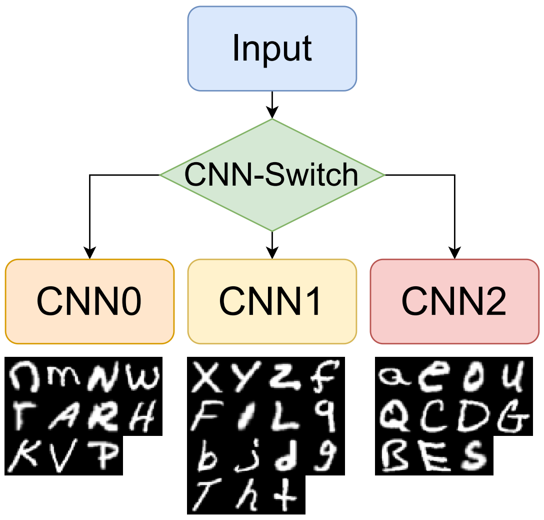


Figure 4.10: CNN tree architecture using dynamic model swapping on SCAMP in which each CNN performs a comparatively more straightforward task using the 4-layer CNN. These four categories are using EMNIST merged classes.

categories of these 37 letters were determined using k-means clustering via Principal Component Analysis [184] shown in Figure 4.11.

With the proposed CNN tree where each CNN uses 2 convolutional layers + 2 fully-connected layers, better accuracy of 86.74% is obtained compared to a single neural network (Table 4.1). As can be seen from Figure 4.2, a third convolutional layer is challenging to extend to improve the CNN performance because the size of feature map is 8×8 after two max-pooling (2 and 4) which is too tiny to add another max-pooling for the third convolutional layer. An alternative would be more fully-connected layers,

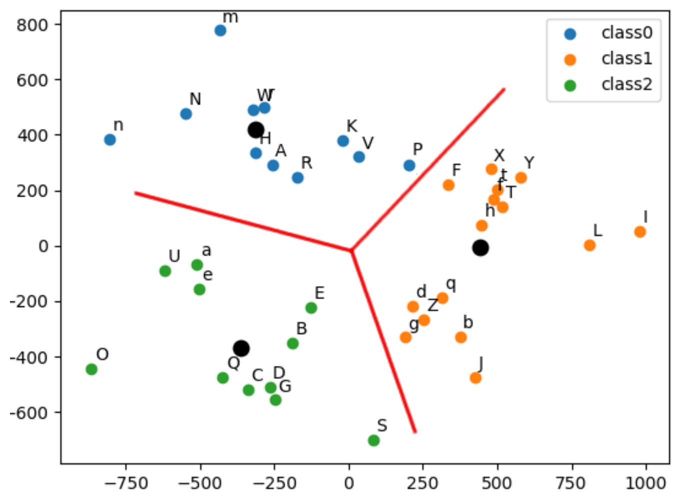


Figure 4.11: 37 English letters are clustered into three categories with k-means clustering. Black dots represent the centre of each category.

Table 4.1: CNN classification accuracy among different CNNs.

| CNNs | Accuracy | parameter amounts |
|---|---|---|
| CNN switch | 95.55% | 132,873 |
| CNN 0 | 94.77% | 133,153 |
| CNN 1 | 83.70% | 133,293 |
| CNN 2 | 93.77% | 133,153 |
| Overall | **86.74%** | – |
| Single 4-layer CNN | **84.20%** | 134,063 |
| Single 4-layer CNN group=1 | **84.55%** | 149,423 |

which is also limited by the hardware resources. To improve the overall classification accuracy, a CNN tree with 4 CNNs (Figure 4.10) is proposed, where the basic idea is to use a combination of four 4-layer neural networks uploaded into SCAMP in sequence to get a closer accuracy with a deeper neural network that exceeds the SCAMP hardware storage/computation capacity. It might be challenging to store all the weights for a deeper neural network on a embedded system, but the parameters in each branch can be uploaded into system according to the last inference result, which alleviates the storage pressure and, in the meantime, obtains a better accuracy than a single neural network. In terms of the CNN tree training, four CNNs are trained separately with the same neural network structure. The accuracy for each CNN can be seen from Table 4.1. In addition, Figure 4.12 shows the binary training process and comparison among single

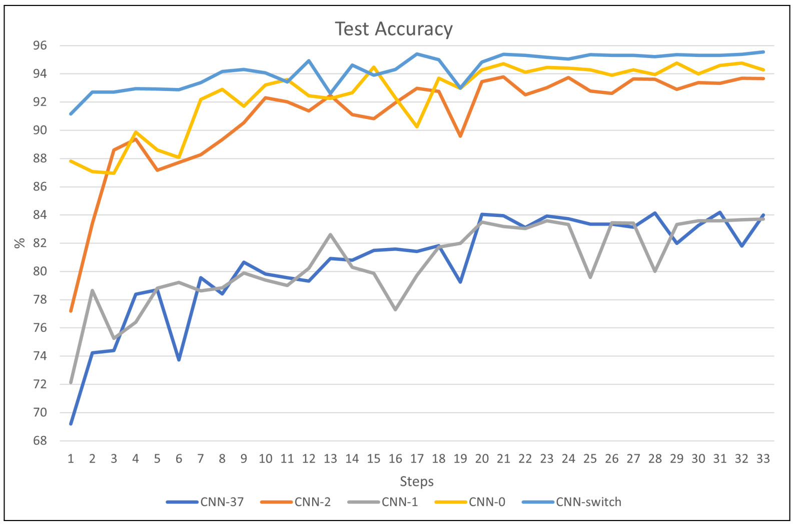


Figure 4.12: CNN tree training performance. We compared the CNN tree with a single CNN scheme in terms of accuracy. With the same binarised CNN architecture of two convolutions and two fully connected layers, the overall accuracy of the CNN tree is better than a single CNN. Although CNN tree sacrifices the efficiency by running multiple CNNs sequentially, it provides a solution to combine several networks for a comparatively complicated task.

network and a CNN tree. Note that, CNN-1 suffers from a poorer performance compared to its counterparts, which results from not only the number of classes (15 vs. 11) but also the amount of similar classes, such as *F and f*, *L and I*, *g*, *q* shown in Figure 4.11.

To evaluate the CNN implementation on SCAMP, Figure 4.5 compares the first fully-connected neuron values between SCAMP and Simulation on PC because errors are mainly caused by analogue signal processing on the PPA, and there would be no error introduced to the last fully-connected layer afterwards since it is implemented on the micro-controller. As shown in Figure 4.5, the neuron values from SCAMP using digital summation is close to the ground truth in simulation and the average absolute errors measured is 22.6 for each neuron. Figure 4.14 shows the prediction results of the proposed CNN tree neural network. The final measured accuracy with EMNIST datasets on the PPA is around 82%, which sees a 4-5% accuracy gap from the groundtruth in the PC simulation.

## 4.7 Conclusion

To efficiently embed artificial neural networks onto the emerging but resource-constrained PPA sensor, we propose a series of novel deployment methods, including purely binarized

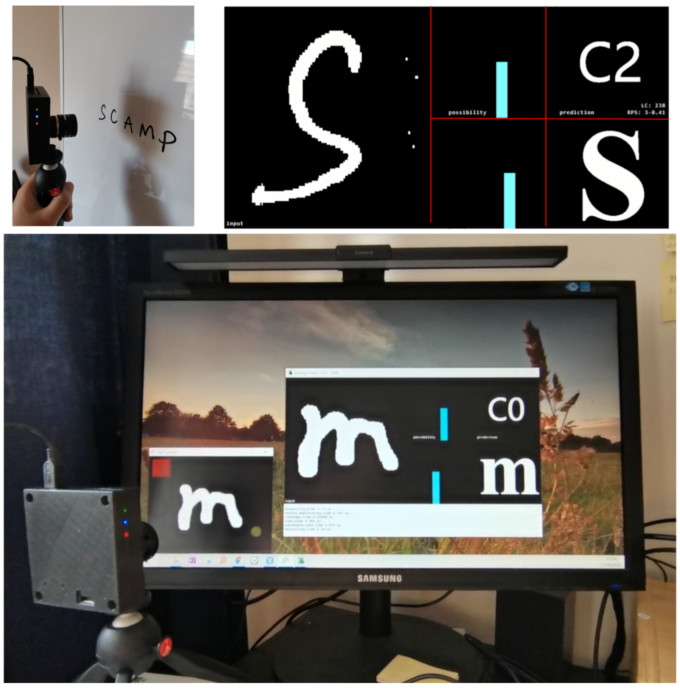


Figure 4.13: A live demo by facing a SCAMP to the handwritten letter on a whiteboard and screen drawing pad. Top left, facing the SCAMP to a whiteboard and the inference results on the top right. Bottom: facing the SCAMP to a handwritten letter on a drawing pad (bottom left). More details can be seen from the experimental video https://youtu.be/8V9vXhXw8X8.

networks, batch norm, trainable adaptive binarization thresholding, group convolution, fully-connected layer with digital summation, and network tree with dynamic swapping scheme for classification, localisation, and segmentation tasks. By combining these strategies, a deeper neural network with a greater capacity for inference is built on the SCAMP, enabling the execution of more difficult tasks. We test the visual competencies of region segmentation and target object localisation with a latency of 3.5 milliseconds for each inference using real pixel processor array (PPA) hardware. Additionally, this work makes use of a novel CNN tree architecture by sequentially running many neural networks based on their outputs. Each network in the tree makes full use of the embedded device's hardware resources. This approach enables the use of a deeper and wider CNN tree on SCAMP and other embedded devices. Two experiments demonstrating the effectiveness of the proposed binarised CNN and implementation method on the SCAMP vision system demonstrate the effectiveness of the proposed binarised CNN and implementation method on 37 letter classification using a network tree architecture and object coarse segmentation using a shared convolution layer.

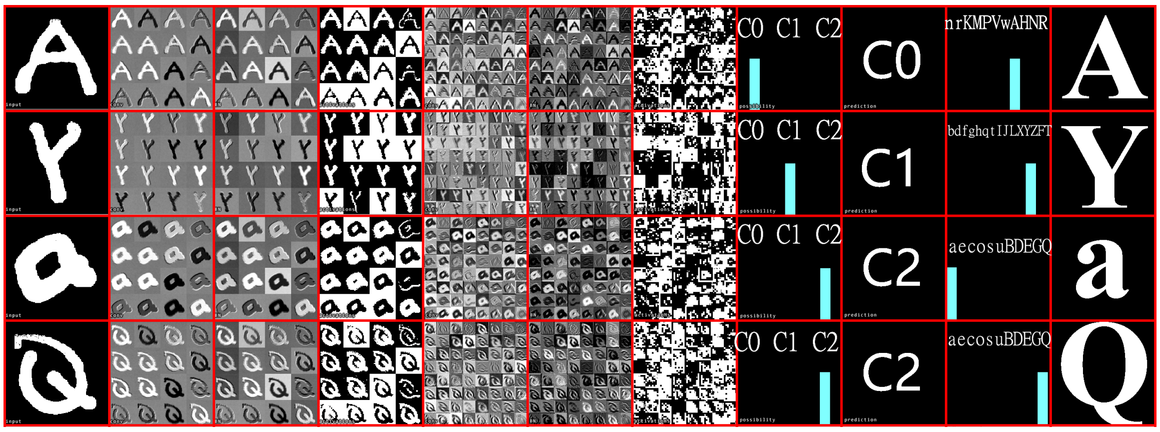


Figure 4.14: Examples of CNN tree inference on SCAMP-5d. For each letter from left to right: (1) Real-time hand-written input image by facing the SCAMP to a writing pad. (2) first convolutional layer. (3) convolutional layer after batch norm. (4) binary feature maps after maxpooling and activation function. (5) second convolutional layer. (6) convolutional layer after batch norm. (7) binary feature maps for fully-connected layer after maxpooling and activation function. (8) prediction bar for first CNN inference results (Category 0, 1, 2). (9) Switch CNN inference results. (10) prediction bar for the second CNN inference results. (11) final CNN inference results. Notice that for each letter inference process, visualisation is only for one CNN.

Table 4.2: Computation time breakdown CNN of single branch

| Processing Steps | Approx. Time Cost ($\mu s$) |
|---|---|
| Imaging and thresholding | 35 |
| Character resize and duplication | 359 |
| 1st Image convolution | 184 |
| 1st Batch norm and activation | 235 |
| 2nd Group convolution | 184 |
| 4×4 maxpooling | 34 |
| 2nd Batch norm and activation | 235 |
| 1st fully-connected layer | 4318 |
| 2nd fully-connected layer | 11 |
| Total time cost | 5595 (178 FPS) |
| number of weights | ≈ 133k |
| power consumption | < 2 W |
| model size | ≈ 0.127 KB |

CHAPTER 5

# On-Sensor Binarized Fully Convolutional Neural Network

Both Chapters 3 and 4 focus on the visual classification inference with binarised neural networks on the focal plane of the SCAMP vision system. However, these proposed neural network architecture struggles to deal with coarse segmentation or localisation problems. In addition, object detection tasks usually require more computation and memory than image classification. Therefore, this chapter trained binarised fully convolutional neural networks and deployed them on the sensor for image coarse segmentation and object localisation. Similar to Chapter 4, where dynamic neural networks are utilised, this chapter entirely takes advantage of the hardware resources of SCAMP and its PPA for multiple network execution on the sensor for diverse purposes.

## 5.1 Introduction

This chapter presents a method to implement fully convolutional neural networks (FCNs) on Pixel Processor Array (PPA) sensors and demonstrates coarse segmentation and object localisation tasks. We trained binarised FCN for both binary weights and activations using batchnorm, group convolution, and learnable threshold for binarisation, producing networks small enough to be embedded on the focal plane of the PPA, with limited local memory resources, and using parallel elementary add/subtract, shifting, and bit operations only. We demonstrate the first implementation of an FCN on a PPA device, performing three convolution layers entirely in the pixel-level processors. We use this architecture to demonstrate inference generating heat maps for object segmentation and localisation at over 280 FPS using the SCAMP-5 PPA vision chip.

For the first time, Liu *et al* [66] proposed a fully convolutional neural network based on the binarised neural network with three convolutional layers. Their work implements object localisation and coarse segmentation based on heat map extraction. In addition, group convolution is utilised to simplify the computing and reduce the intermediate parameters.

Fully convolutional neural networks (FCN) have been used across many modern computer vision tasks such as object detection [185], classification [186] and segmentation [187, 188]. However, the deployment of deep FCN usually relies on powerful GPU/CPUs, which are typically not present in emerging embedded edge devices, where cost and energy considerations dictate stringent limits on storage and computing resources. Despite this, there is an ever-increasing demand for artificial intelligence on such edge devices. One promising approach to edge computing hardware is represented by Pixel Processor Arrays (PPA). Unlike conventional vision systems, which consist of separate sensing and computing hardware, PPA devices are emerging vision architectures, integrating sensing, storage, and computing on a single silicon chip (Figure1.2) [10, 189]. Such integration optimises data movements in the system, promising high performance and low-power consumption, but requires careful algorithm implementation to efficiently utilise the hardware resources available in an on-sensor computing device. This paper demonstrates how to implement and deploy binarised FCNs on PPA hardware.

Networks with binary weights and activations are challenging to train and usually suffer from performance drop compared to their floating-point equivalent. The use of batch normalisation has been proposed to avoid gradient explosion and train binarised neural networks successfully [176]. In this work, we introduced batch normalisation into binarised FCNs to improve the training efficiency and inference performance on PPA arrays. We implemented a purely binary convolutional network containing both binarised weights and activations. The use of binary activations alleviates accumulative errors introduced by approximate computations used to perform image convolutions upon PPA hardware devices [171]. This error mitigation allows our approach to performing deeper networks than previous work [64, 65, 110] wholly upon the focal plane without encountering an increasing loss of accuracy that would occur otherwise. Furthermore, it is noted that the implementation of the batch normalisation, the sign activation function, and learnable activation threshold for binarised activations is equivalent to adding a bias matrix to the layer activations, which significantly simplifies the inference process on the sensor. With this binarised FCN, the inference calculation process can be implemented entirely with efficient add/subtract, threshold, and shifting operations for all layers. This scheme benefits explicitly PPA computing devices such as the one shown in Figure 1.2 because it matches the simple instruction set of these devices and reduces the impact of calculation errors caused by noise accumulation when using analogue registers (AREG) for storage and calculations, especially for activations. For experiments, we trained binarised FCN and deployed it on the PPA for object localisation and coarse segmentation.

The concept of on-sensor computing originates from emerging novel circuit designs that enable direct signal processing on the sensing chip [15]. The SCAMP vision sensor [39] used in this work is based on a PPA concept implemented using mixed-signal analog/digital datapath (Figure 1.2), other devices such as SONY IMX500[1] or Aistorm Mantis[2] integrate sensing and computing resources in a single device using alternative strategies. Our work uses the SCAMP analogue/digital PPA to implement a binarized FCN efficiently.

The main contents of this chapter can be summarised as we propose, train, and demonstrate the use of a purely binarised network (both binary weights and activations) specifically for PPAs. This approach of binary activations addresses the accumulation of analogue computing errors and value saturation after each layer, thus enabling deeper networks while maintaining performance. Then, we present the first implementation of an FCN architecture for PPAs. Our approach uses group convolutional layers [181] and stores hundreds of convolutional filter weights upon the focal plane of the PPA. Thirdly, unlike earlier work, we applied batch normalisation during training and utilised this to learn bias parameters applied during inference on the PPA device. Lastly, we first demonstrate object localisation and coarse segmentation tasks on a PPA, with previous works only concerned with classification tasks. In addition, we created a simulated environment and datasets to support PPA developers and researchers for idea validation off and on-sensor.

[1]https://developer.sony.com/develop/imx500/
[2]https://aistorm.ai/mantis-2/

## 5.2 Related Work

Recent work for PPAs has concentrated on CNNs and demonstrated classification tasks. However, we found no previous work on FCNs, which are essential for further tasks like localisation and segmentation. The recent research progress of the neural network on the SCAMP PPA has been illustrated in section 1.6.2.

**FCN on other unconventional sensors:** The difference between our work and these papers mentioned above lies in our proposed new network architecture of three on-sensor convolutional layers with binary weights and activations. Furthermore, to train such a neural network that is highly adaptive to the PPA, batchnorm is utilised, enabling new segmentation-related applications. There has been several work on CNN for classification tasks [64, 65, 110] but few research or implementation of FCN on the unconventional sensors, which might result from a higher computation and storage demand for FCN than CNN. However, our work is the first to propose a method to implement FCN on the PPA for segmentation tasks that can not be implemented with earlier CNN architecture.

## 5.3 Binarized Fully Convolutional Neural Network with Batch Normalisation

Neural network architectures for PPAs must be carefully designed taking into account model size, architecture, and the feasibility of exploiting the PPA's parallel computation and on-sensor storage, which is essential due to the limited on-sensor resources compared to standard computer hardware which may have access to powerful GPU/CPUs. This section attempts to find a balance between the neural network performance and its efficient implementation on the PPA.

### 5.3.1 FCN architecture on sensor

FCN is a CNN architecture providing pixel-level classification, targeting image segmentation [187]. This paper proposes a 3-Conv layer FCN that can be implemented on a PPA sensor. This work extends from previous CNN classifications that were done using fully-connected output layers [64, 65]. This paper uses FCN for heat map generation by adding one convolutional layer with 128 filters and replacing the final fully-connected layer with a convolutional layer of kernel size $1 \times 1$. Figure 5.1 shows the overall FCN

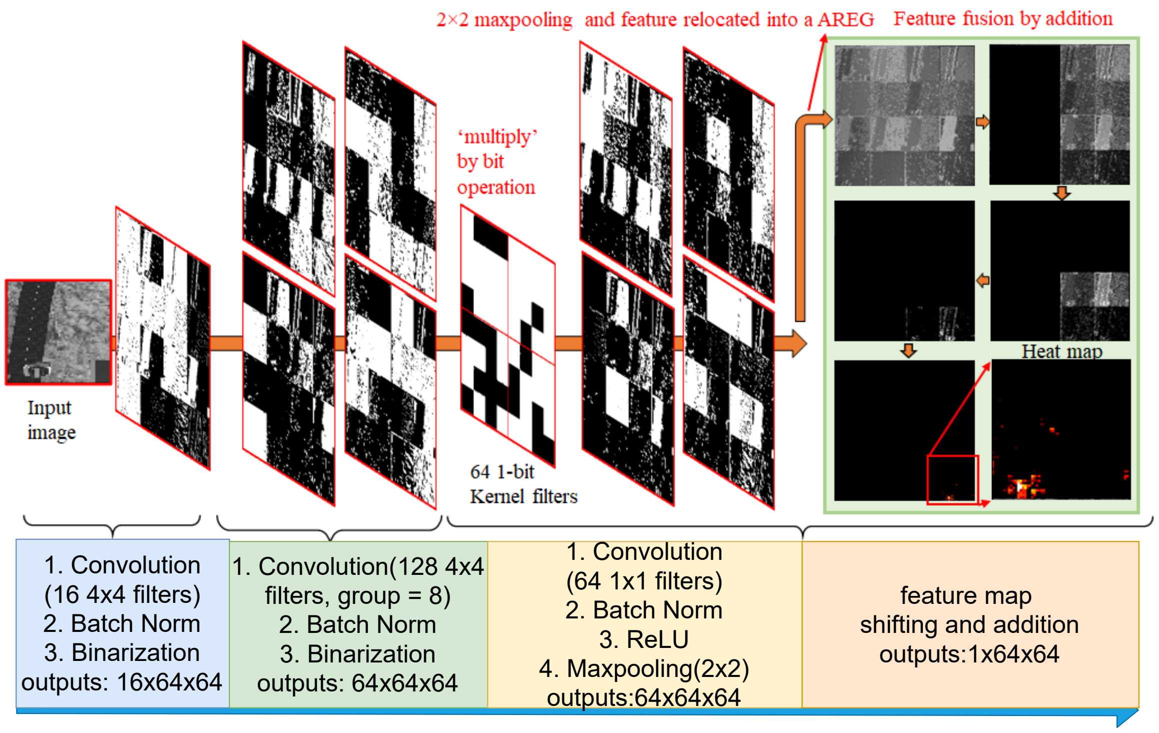

2×2 maxpooling and feature relocated into a AREG
Feature fusion by addition
'multiply' by bit operation
Input image
64 1-bit Kernel filters
Heat map
1. Convolution (16 4x4 filters) 2. Batch Norm 3. Binarization outputs: 16x64x64
1. Convolution(128 4x4 filters, group = 8) 2. Batch Norm 3. Binarization outputs: 64x64x64
1. Convolution (64 1x1 filters) 2. Batch Norm 3. ReLU 4. Maxpooling(2x2) outputs:64x64x64
feature map shifting and addition outputs:1x64x64

Figure 5.1: An overview of an on-sensor FCN architecture and inference process using a PPA for heat map generation. A three-layer FCN architecture is used in our work. The first convolutional layer can be seen from Figure 5.4 in detail. In the second convolutional layer, 128 convolutional kernel filters are applied upon the 16 input binary feature maps from the first layer, generating 64 feature maps with a convolution group setup of eight. The fusion of intermediate extracted features is implemented by addition within each group. The third layer uses binary filters with a size of $64 \times 1 \times 1$, hence the final feature maps can be obtained by 'multiplication' with bit operation based on DREG. The final heat map is generated by combining these input 64 feature maps by shifting and addition operations.

architecture with configurations for each layer. Figure 5.4 illustrates the first convolution layer, generating 16 binary feature maps. The second layer adopts group image convolution [190] of 8 on the input 16 feature maps to make a trade-off between convolution computation complexity and network performance, where each of 64 outputs is generated by adding two intermediate feature maps (Figure 5.8). These 64 binary feature maps from the second convolutional layer are stored in 4 DREG. The third layer generates the final heat map representing the prediction probability distribution, taking these 64 binary feature maps and combining them within an AREG. Each binary feature map is multiplied by an associated weight of -1/1.

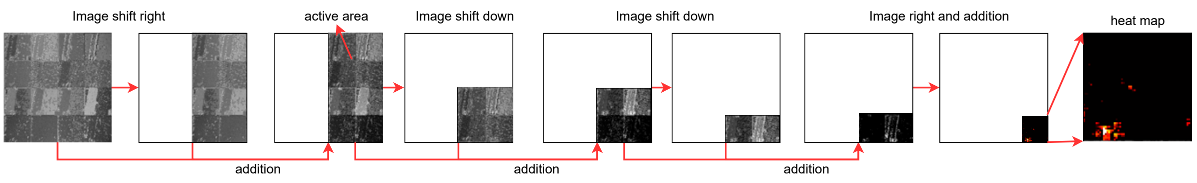


Figure 5.2: Feature map shifting and addition process: the final heat map is generated by adding the feature maps from the third layer.

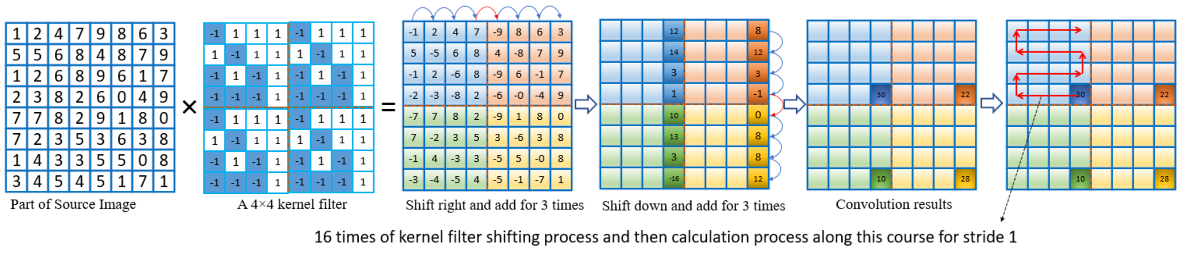

Part of Source Image
A 4×4 kernel filter
Shift right and add for 3 times
Shift down and add for 3 times
Convolution results
16 times of kernel filter shifting process and then calculation process along this course for stride 1

Figure 5.3: Image convolution on PPA with sign inversion, bit-shifting, and addition.

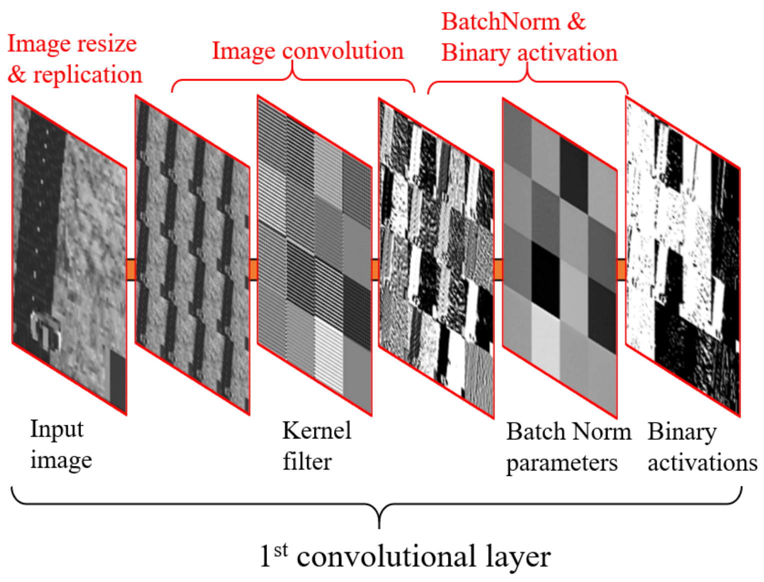


Figure 5.4: First convolutional layer of the FCN.

## 5.3.2 FCN deployment on the PPA

This section gives the implementation detail of the binarized FCN on-sensor. Figure 5.9 shows the layouts and operations of input 16 binary feature maps, 128 kernel filters, 64 generated feature maps.

**First Layer:** 16 binary filters are replicated to fill a DREG (Figure 5.5) for parallel convolution purpose [64, 65]. Figure 5.3 shows the image convolution process on the PPA.

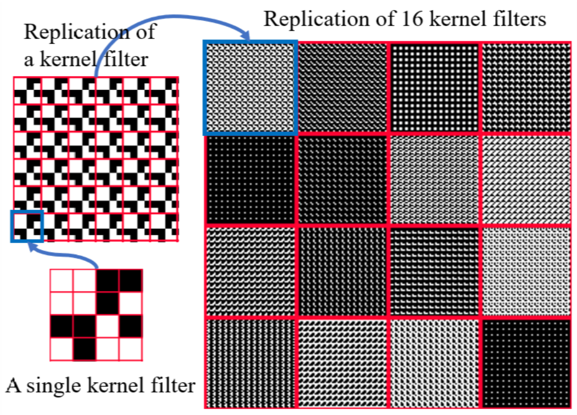

Replication of a kernel filter
Replication of 16 kernel filters
A single kernel filter

Figure 5.5: The layout of 16 binarized convolutional kernels in a DREG for the 1st layer.

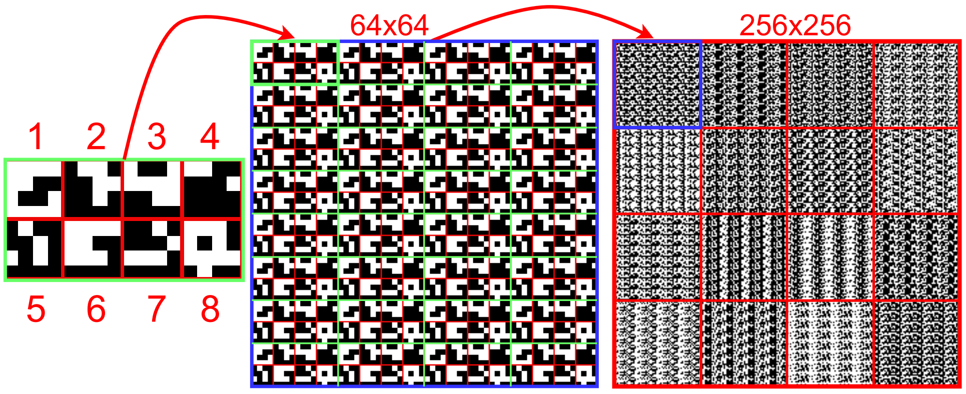


Figure 5.6: Storage structure of filters on DREG. Left: The layout of 8 filers (128 filters in total) in a block of size 8×32. Middle: The layout of 8 kernel filters stored in one 64×64 block after replication. Right: The layout of 128 kernel filters in a 256×256 DREG.

The image convolution on the PPA can be decomposed as 'multiplications', shifting and addition and the convolution result are obtained by performing the shifting and addition process 16 times with a stride = 1. Then the pre-calculated bias $B$ is plotted into 4×4 grids on a AREG and is subtracted from the feature map (seen in Figure 5.4 ). Then the output binary image is obtained by binarizing the feature map after subtracting $B$. In this layer, $tanh$ is used as the activation function. When implementing inference on the sensor, the $tanh$ activation function is transformed into binarisation with a sign function, with offset computed from batch norm parameters as can be seen from Equation 4.4. This layer shares some similarities with work [65] including the image resize, replication, and image convolution but with an extra activation binarization process.

**Second Layer:**

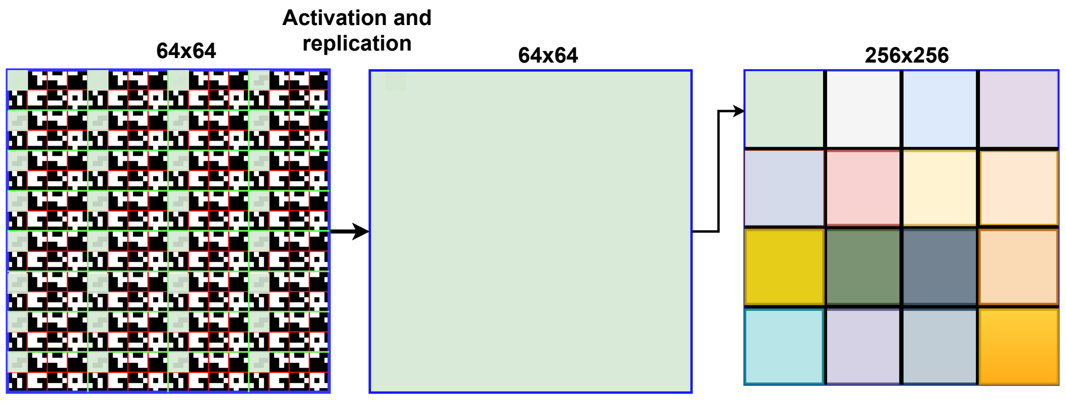


Figure 5.7: 128 kernel filters can be used by activating and replicating different areas. Left: A kernel filter is activated and then replicated to fulfil a 64×64 block (right). Right: 16 kernel filters in a DREG after replication.

The key to the second layer is to implement the group convolution with 16 feature maps as inputs and 64 as outputs. By dividing them into eight groups, thus there are 128 binary filters need to be stored on the sensor. The layout of filters directly affects inference efficiency. We design a storage structure for filters in first (Figure 5.5) and second layer. Figure 5.6 illustrates the layout of these filters within one DREG for efficient reading out and then performing convolution. Notice that only the right image in 5.6 is stored in one DREG, and the filters' layout can be generated offline. As can be seen, each time before performing a convolution operation, the corresponding 16 kernel filters are activated in parallel, shifted, and replicated to fill each 64×64 block in 256×256 PEs (Figure 5.7). 16 kernel filters can be replicated to be stored in a DREG each time. This filter storage structure can also extend to store more filters, following the similar way to fill all PEs with 16 filters.

In Figure 5.8, to implement a second convolution layer with 8 groups, the input 16 binary feature maps are first transformed by switching position of adjacent maps, which is followed by convolution with associated 32 filters for these 32 feature maps. Then 16 gray-scale feature maps are obtained by adding each two of 32 maps. By performing convolution for another 96 filters, 64 gray-scale feature maps can be derived. The bias matrix is subtracted and then after binarization, 64 binary feature maps are generated.

**Third Layer:**

In this layer, as shown in Figure 5.1, 64 1-bit filters are plotted on a DREG, followed by 'multiplication' with the 1-bit feature maps from the previous layer. After 1×1 convolution, these 64 feature maps in 4 DREG is relocated to 1 AREG after $2 \times 2$ maxpooling. The

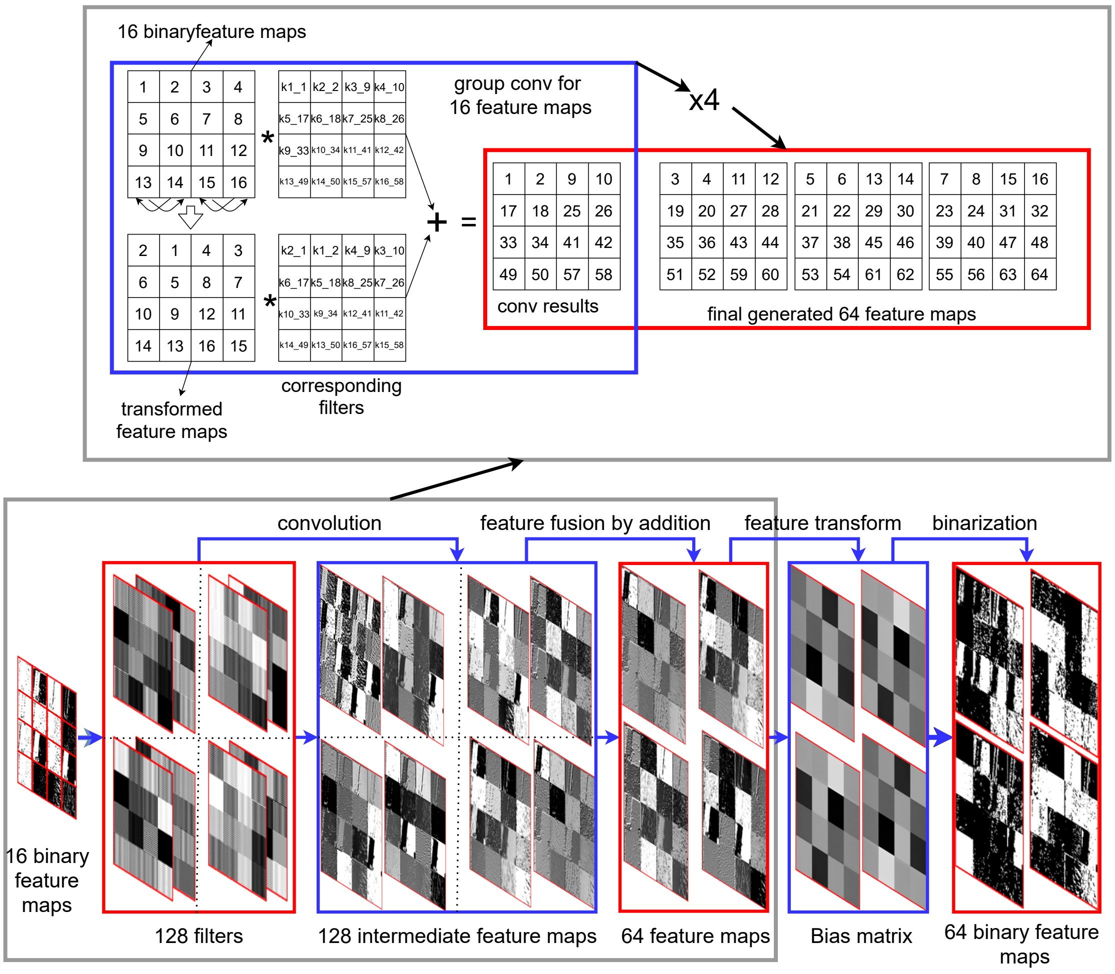


Figure 5.8: Group convolution on the PPA for the second layer. Top: schematic diagram of group convolution. Numbers within grids represent the index of feature maps and filter layouts. Bottom: group convolution on the sensor. 128 convolutional kernel filters are applied on 16 input binary feature maps (far left), generating 64 feature maps by adding each two of 128 intermediate maps. Then these 64 maps are subtracted by a bias matrix, followed by a binarization. Finally, 64 binary feature maps are obtained for the next layer.

summation of these extracted features can be obtained by shifting and adding into one $64 \times 64$ heat map. Unlike in the previous two layers, the activation function for this layer is ReLU to generate a gray-scale feature map as the final prediction result of the network.

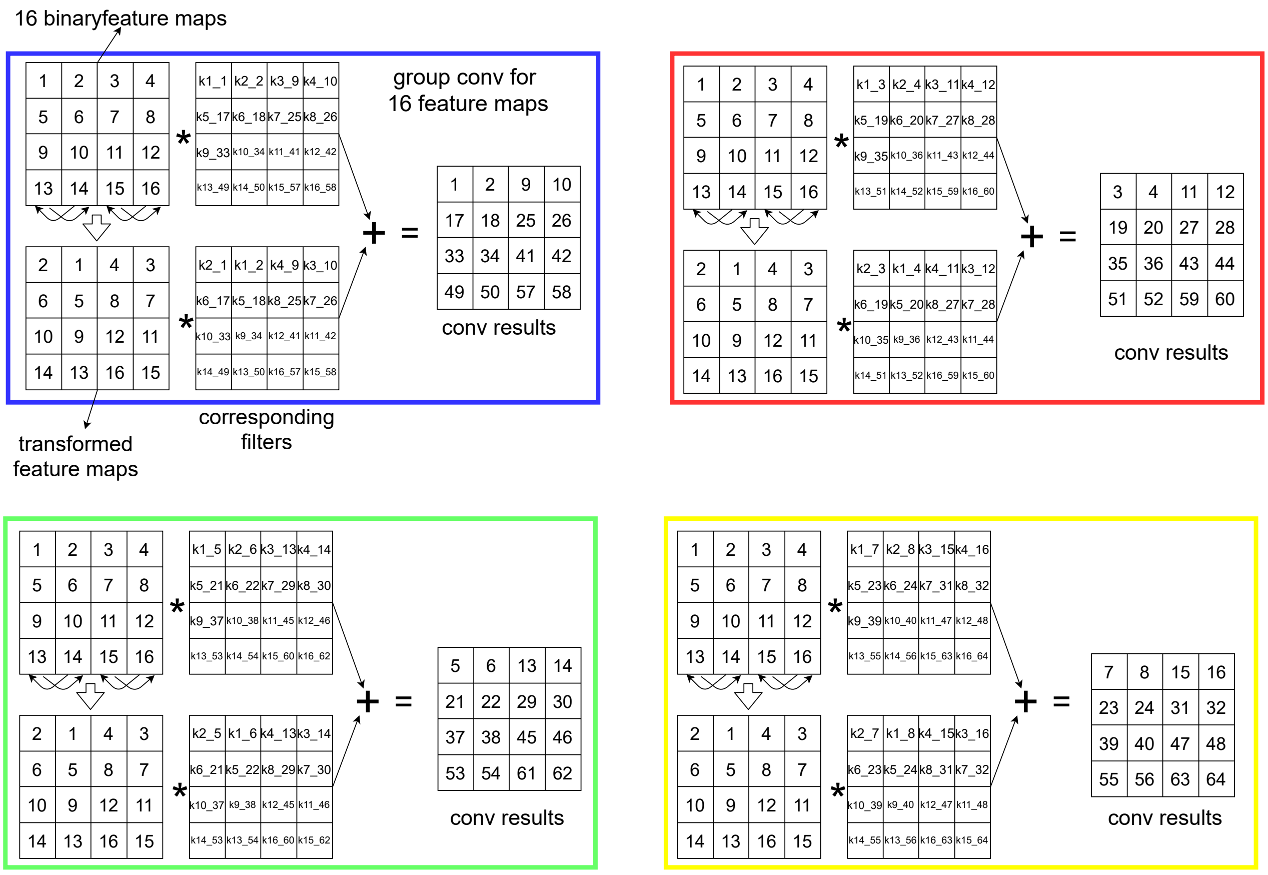


Figure 5.9: 64 feature map generation on sensor after group convolution.

### 5.3.3 SCAMP-5 Inference, Experiments, and Evaluation

This section demonstrates the application of the proposed network architecture to coarse segmentation and object 2D localisation from a bird's eye view on a real SCAMP vision system (Figure 5.10). We set up a realistic environment in Webots[3] [191] robot simulator (Figure 5.11) for data collection and the validation of FCN deployment on the real sensor. Training, testing and validation datasets are collected by repeatedly taking images from a flying drone equipped with a simulated "SCAMP" and then validation images are sent to the real PPA for inference. Binarized FCN is trained offline based on these datasets with the method proposed in Chapter 3. The whole neural network for both coarse segmentation and localisation is performed on the sensor.

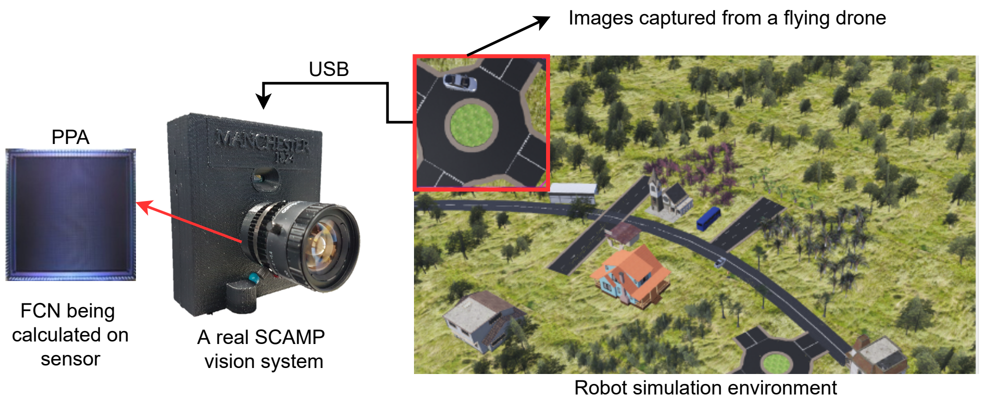

Images captured from a flying drone
USB
PPA
FCN being calculated on sensor
A real SCAMP vision system
Robot simulation environment

Figure 5.10: Experimental setups using a robot simulator and a real SCAMP vision system where the neural networks are fully computed on the sensor.

| Task | IoU |
|---|---|
| Road segmentation on computer | 74.0% |
| Road segmentation on sensor | 69.3% |
| Grass segmentation on computer | 76.6% |
| Grass segmentation on sensor | 72.9% |

Table 5.1: IoU performance comparison between simulation and on sensor for coarse segmentation.

### 5.3.4 Coarse Segmentation

In this experiment, synthetic images are collected randomly from a bird's view of a drone in the simulator and then stored on the computer. Then these collected images are divided into three groups: training, testing, and validation datasets. After training on the computer with the training and testing data, the neural network model is implemented on the SCAMP with the trained model. Finally, the validation images are sent to the SCAMP one by one for the inference performance evaluation. Since the image capturing process is offline and in the absence of full-loop control like in Section A.1.2, there is no time constraint on image capture.

Figure 5.11 shows the samples of collected datasets and their annotations for segmentation of road and grass. To validate the performance of the proposed network on different tasks, a road and grass coarse segmentation is explored in this section. As shown in Figure 5.11, we directly use the road/grass shape as the ground truth for coarse segmentation. Notice that the trees and grass areas often share similar gray-scale levels with the road, making coarse segmentation unfeasible by simply using binary thresholding. Tab. 5.1 shows the Intersection over Union (IoU) performance comparison between FCN

[3]https://cyberbotics.com/

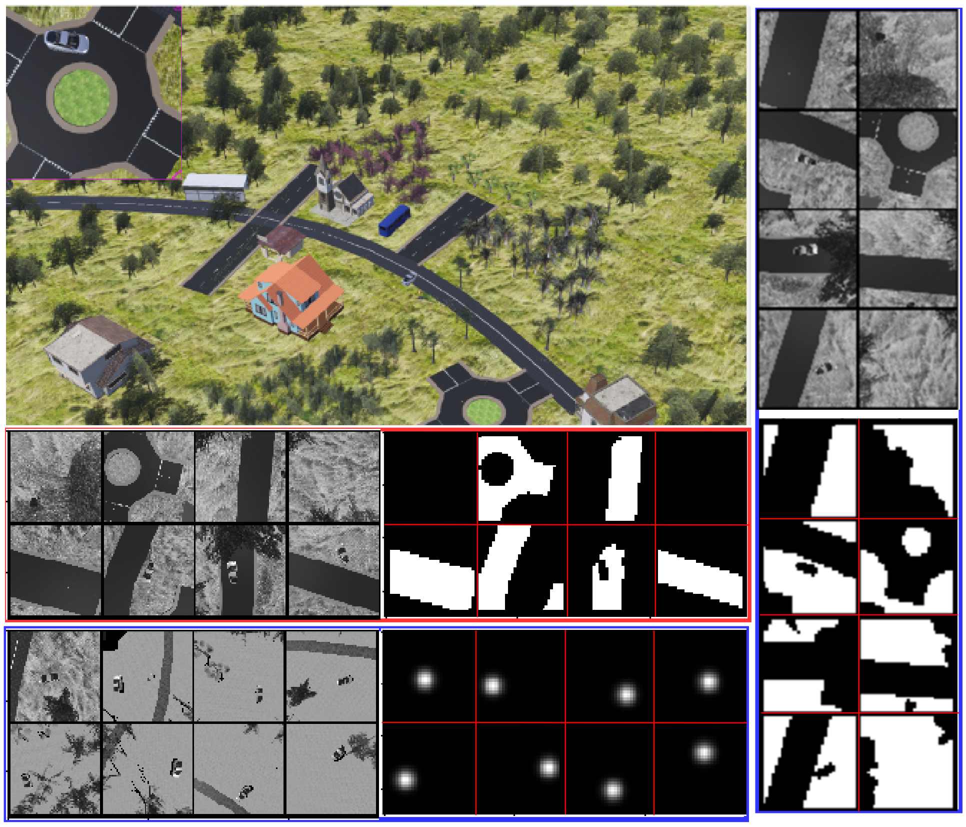


Figure 5.11: The data collection environment where a vehicle moves around and images are generated from a bird's eye view of a simulated drone. Top left: A robot simulator for environment setup and collection of training data. Middle left: The collected images from the drone's camera are converted into gray-scale images for the PPA and the segmentation annotations of the road. Right: The training and annotated datasets for grass segmentation. Bottom left: the Gaussian distribution is used to represent the vehicle position within an image.

inference on simulation and on sensor. Specifically, IoU is measured here by counting the number of intersected pixels over the number of united pixels of the predictions and groundtruth. In addition, Figure 5.13 compares the FCN training process for segmentation task between using and not using batch norm, which shows the binarized FCN dose not converge without batch norm, justifying its use here. Some of the results can be seen from Figure 5.12. Table 5.1 compares the experimental results on sensor and its counterpart baseline on computer with identical neural networks and validation images. More results can be seen from the experimental video https://youtu.be/Z_ydv_0DRnM.

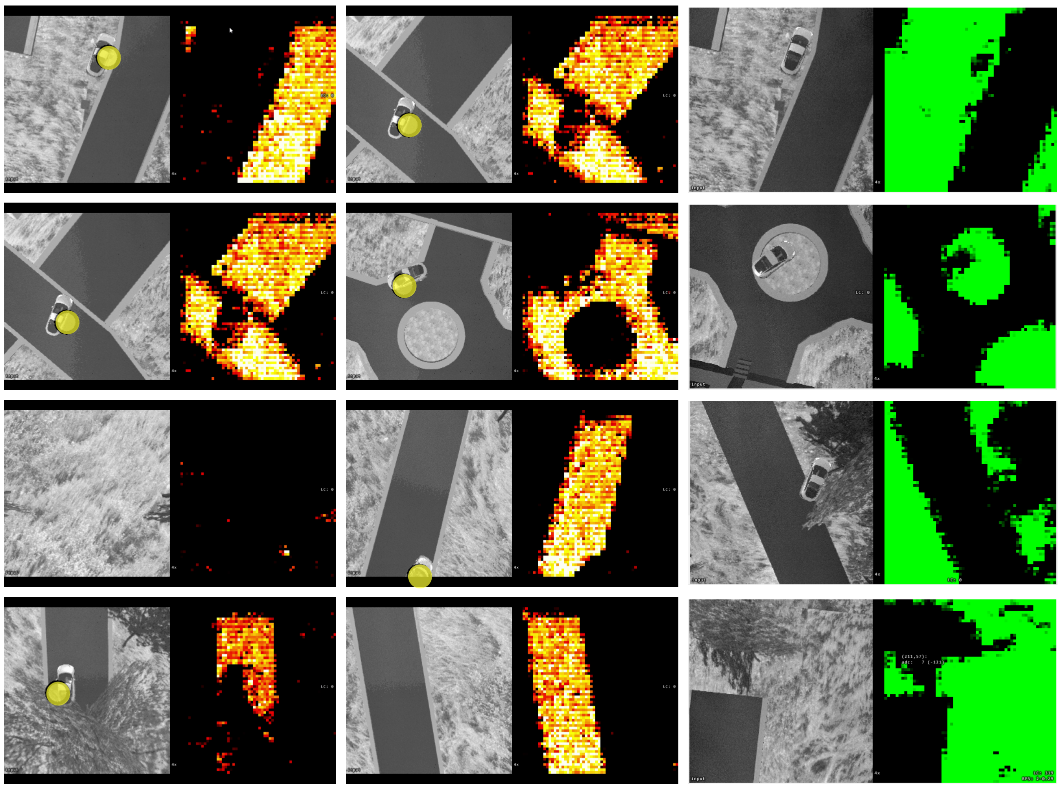


Figure 5.12: FCN inference results on-sensor. The left column is sensor's input grayscale image with yellow dots indicating the FCN inference localisation prediction. The right column is the inference results for coarse segmentation on sensor. The density and distribution of colourful points (right) represent the possibility of position of the road (bright yellow) and grass (green) segmentation. The experimental performance for localisation (accuracy) and segmentation (IoU) on the PPA can be seen in Figure 5.16 and Table 5.1

### 5.3.5 Object Detection

We also implemented an object detection task based on the heat map. As for the object localisation, rather than using the vehicle segmentation image as the ground truth for training, we use Gaussian position distribution (Figure 5.11) as the ground truth since the probability distribution is adequate to represent the object 2D localisation and the current performance of the on-sensor shallow binarized neural network is not capable of generating grain-fine segmentation of the car. For the validation, a distance threshold is set from 0 to 63 to count the number of predictions with a distance to the groundtruth that falls into this threshold. A zero distance means a perfect prediction. The final localisation is obtained by the weighted sum of all the possible positions. After the test, within a distance of 10 pixels, the vehicle localisation accuracy for simulation

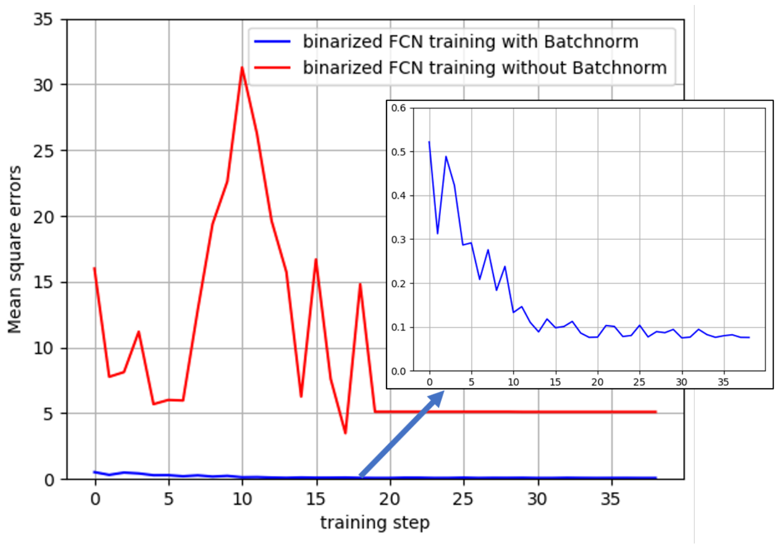


Figure 5.13: Training process comparison for grass segmentation with and without batch norm.

and SCAMP is around 88% and 83% respectively (Figure 5.16). Tab. 5.2 shows the FCN performance in terms of time, power consumption and model size.

### 5.3.6 Pupil detection

Considering the light weight and low-power consumption of the PPA chip and the increasing popularity of virtual and augmented reality (VR/AR) based on the eye movement, it is promising to mount the PPA chip to a wearable device such as glasses in the near future, hence we explore the pupil detection with the proposed binarized FCN based on the public dataset of eye images: TEyeD [192]. Figure 5.14 shows some of the training images and their annotations and Figure 5.15 shows results comparison between SCAMP and simulation inference, where the accuracy curve is plotted according to the Euclidean distance between simulation/scamp inference results and the groundtruth. Within a distance of 10 pixels, the localisation accuracy for simulation and scamp is around 88% and 83%, respectively. More experimental results can be seen from[4].

Notice that there is around 5% - 6% performance gap for the experiment on sensor compared to the simulation. This is due to noise in the convolution operation performed on AREG because of the inherent non-idealities of analog computation [38] and some random bit-flipping errors observed in DREG when performing massively parallel shifting and replications. Mitigation of these issues requires further software or hardware

[4]https://youtu.be/APMixrUOT80

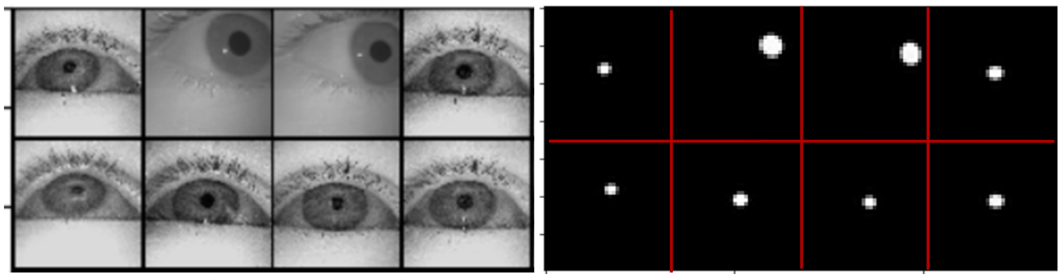


Figure 5.14: Selection of pupil images (Left) and their annotations (Right) from TEyeD datasets.

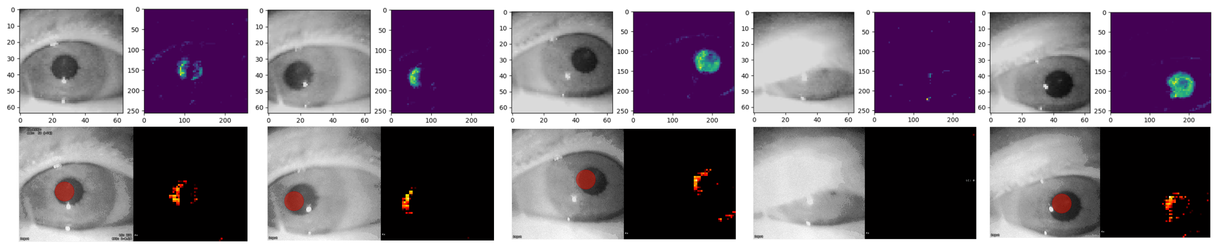


Figure 5.15: Selection of inference results comparison between FCN with off-sensor simulation with a computer (top) and on-sensor PPA (bottom). The red dot at the bottom left of each frame is the final position prediction calculated from the heat map on the right.

solutions. In this work, we tried to find a balance between network complexity and viability for deployment upon the available PPA prototype hardware. Pixel-wise accurate segmentation, with a quality equal to one that can be obtained using a CPU/GPUs hardware, using embedded low-power SCAMP-5d vision system, is still a challenging task with current hardware and neural network architecture.

## 5.4 Shared Convolutional Layer for Multiple Tasks with FCN

We use dynamic model swapping strategy to run these three networks on the PPA by sharing the first convolutional layer among these networks (shown in Figure 5.17) considering the identical testing environment and similar network structure. In that case, less storage requirement for DREG and higher inference efficiency without uploading extra weights from the FLASH can be obtained for three networks. Rather than train these three networks concurrently which might result in unbalanced training process for each task because of their different scale of loss [193]. We adopt a straightforward but

| Processing Steps | Time Cost ($\mu s$) |
|---|---|
| Image replication | 112 |
| 1st Image convolution | 184 |
| 1st Batch norm and activation | 235 |
| kernel filter activation and replication | 187 |
| 2nd Group convolution | 184×4 =736 |
| 2×2 maxpooling | 35×4 = 140 |
| 2nd Batch norm and activation | 235×4 = 940 |
| Third convolutional layer | 966 |
| Total time cost | 3500 (285 FPS) |
| number of weights | 2,578 |
| power consumption | ≈ 1.5W |
| model size | ≈ 0.31 KB |

Table 5.2: Computation time, performance and weights for heat map generation with the binarized FCN on sensor.

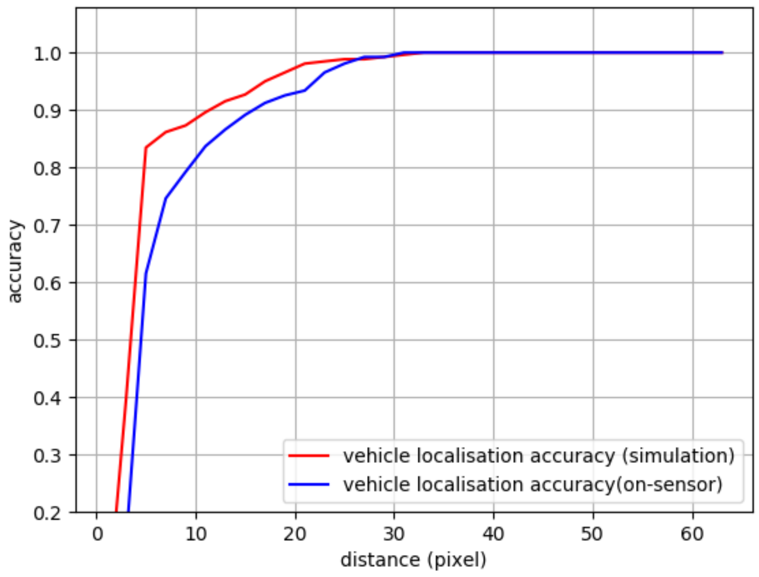


Figure 5.16: Performance comparison between simulation and on sensor for localisation task.

efficient method by training neural network for localisation first, locking the weights in the first convolutional layer and then train the other segmentation networks respectively. The reason to train these three networks is that the localisation task is more sensitive to the final heat map prediction hence better to train separately, while segmentation for road and grass are similar tasks which are more tolerant to the final heat map prediction. Some selected results can be seen from Figure 5.12. And a break down of time cost within

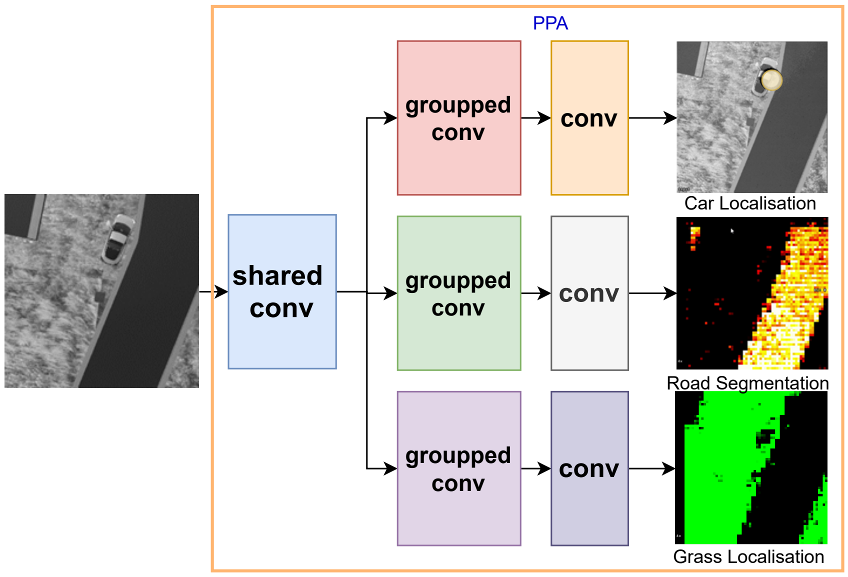


Figure 5.17: A FCN tree architecture with a shared convolutional layer for three tasks.

one branch of networks for each inference step can be seen from Table 4.2. Notice that, according to our test, it takes around 26 milliseconds to upload a CNN model from the FLASH into a DREG. However, this issue can be solved by pre-storing the binary models into DREGs or AREGs before the CNN inference process.

## 5.5 Conclusion

On-sensor computing is important for embedded and low-lag, low-power vision systems. Due to their compactness and computational advantages, binary FCNs are increasingly appealing. This paper proposes and implements a method that demonstrates carrying out an inference with a binarized FCN on an on-sensor computing device. In contrast to previous works that have mainly focused on classification with a fully-connected layer, we, for the first time, exploit the FCN architecture design, implementation method, and inference on the sensor. We validate, using a real pixel processor array (PPA) hardware, on the visual competencies of region segmentation and target object localization with a latency of 3.5 milliseconds for each inference. With the development of the future generation of on-sensor devices in terms of image resolution, manufacturing techniques, and local memory capacity, we believe our proposed binarized FCN can be extended with extra layers for more challenging applications.

# Chapter 6

# Conclusions

This chapter concludes the thesis by examining the advantages and limitations of our proposed methodologies for on-sensor visual inference for robot navigation, object classification, coarse segmentation, and localisation. In addition, possible future research directions are discussed.

## 6.1 Summary of Contribution

This thesis explores a newly but fast-growing notion of on-sensor computing with a series of associated algorithms and potential applications. Based on the output of the research presented in this thesis, our contributions can be summarised as follows:

### 6.1.1 Main Contributions

- In Chapter 2, we explored the robot reactive navigation based on the on-sensor visual information computing guiding the ground vehicle to perform high-speed motion according to on-sensor interpreted information from the pre-set patterns. The RC model car (1:10) can perform reactive behaviour at an average speed of 2.2 m/s with a maximum speed 3.88 m/s. The on-sensor image processing algorithm runs up to 200 fps indoors for this scenario.

- After performing low-level image processing on the sensor for robot-related applications, the author then re-focuses on the higher level on-sensor image inference with the neural network for the more general classification, localisation, and segmentation tasks. Hence, in Chapter 3, the author proposed a new method for parallel image convolution implementation. Then, a series of methods are proposed to deploy the trained binary neural network model consisting of various layers on the SCAMP PPA. Furthermore, the proposed binary neural network and on-sensor implementation method were validated through different classification experiments, including hand gesture recognition, plankton classification and extremely high inference test over binary classification with speed over 17,500 Hz.

- To further deepen the neural network on the sensor and restrain the noises caused by the analogue registers for more sophisticated applications, Chapter 4 proposed the binarised neural network (both binary weights and binary activations). A series of new methods were presented to implement this neural network onto the PPA, mainly including batch normalisation, the neural fully-connected layer with more accurate bit-counting, and the learnable binarisation threshold for binary activations. Moreover, a dynamic neural network model swapping scheme is proposed to deal with classification applications with much more labels. We achieved around 82% accuracy with 178 FPS on 37 alphabet classification with the binarised CNN and dynamic swapping method.
- Chapter 5 implemented the binarised fully convolutional neural network of three layers for coarse segmentation and localisation on the SCAMP PPA. These methods mainly include binarised FCN training, group convolution, and image convolution with the kernel of size $1\times1$. The experiment demonstrates both coarse segmentation and object localisation based on the heat map generation in the experiment. The FCN is tested on both synthetic and real datasets, where the IoU performance on the road and grass segmentation is around 70%, which sees a 5% accuracy drop compared to the same neural network in a computer. Then, the on-sensor pupil detection is tested, resulting in 88% and 83% localisation accuracy within 10 pixels off the sensor and on the sensor, respectively. The FCN inference achieved around 285 FPS with an approximate power consumption of 1.5 $W$.

### 6.1.2 Additional Contributions

- This paper reviews the on-sensor works on PPA conducted in the last 15 to 20 years in Chapter 1, where key algorithms and applications reviewed, providing a comprehensive reference to future researchers on the relevant area.

- In Chapter 2, A proximity estimation for a ground vehicle obstacle avoidance was explored by sending an eclectic mix of features from the focal plane into a recurrent neural network. The non-holonomic RC model car run randomly indoors at an average speed of 1.15 m/s. The work also contributes to the on-sensor mapping and localisation, where the a sequence of in-coming real-time images are filtered by LBP and then stored on the registers of the PPA as the map. By using the motion model information of the PPA and comparing the new images in the same environment where the map was collected with the stored map, the position of this image can be estimated. The F1-Score performance of our proposed localisation method is 93.38% on the public Norland dataset and 82.7% on a simulated environment with a latency of 3.168 ms for a single localisation.
- In Chapter 3, a visual sensory-motor platform was built to directly execute the instructions from on-sensor data without the use of an intermediate control unit. The sensor-action system establishes a direct link between sensing and action, allowing for the complete control of a digital servo motor with a control frequency of 333 Hz.
- A semi-simulated platform is developed where a real SCAMP vision system can be integrated with the Coppeliasim robot simulator.

## 6.2 Discussion

As illustrated above, PPA is a versatile on-sensor computing device designed with the novel electric circuit. The unique hardware design enables varying image processing algorithms ranging from low- and mid-level image processing to high-level neural network inference. These algorithms then allow a wide range of applications in the area of machine vision and robotics. Although our approaches achieve state-of-the-art performance on the focal plane of the SCAMP PPA, the following aspects require further research.

### 6.2.1 Robot Navigation with a SCAMP PPA

The reactive navigation described in section 2.2 is restricted to localising a single predefined object. Hence, autonomous mobile robot navigation using a SCAMP PPA is still a long way off. Nevertheless, in Chapter 2, we proposed a series of methods for robot localisation and navigation and took a step toward a more intelligent robot system through the use of on-sensor computing.

### 6.2.2 Deep Neural Network on the Sensor

The capacity of neural networks proposed in Chapters 3-5 is not sufficient to fit sophisticated classification or fine-grained segmentation tasks like CPU/GPUs do. The reason for this in terms of algorithms is that the proposed neural networks for the SCAMP PPA still adopt conventional neural network architecture, which is more favourable to the conventional CPU/GPU with rich sequential and/or parallel digital floating-point computing supports.

### 6.2.3 Accuracy Drops

There are several performance gaps during the process of neural network binarisation and deployment onto sensor. The neural network is binarised on the computer which experiences the first accuracy loss because of the overt quantisation. In addition, the binarised neural network needs further deployment process to the terminal, resulting in the second performance loss.

### 6.2.4 Discussion on Performance Comparison with Other Platforms

In order to measure the performance of SCAMP vision system against commonly used CPU and GPU platforms, [72] conducted a series of experiments utilising the identical kernel operations for SCAMP, GPU, and CPU. We applied the same SCAMP vision system and similar kernel operations in this thesis, hence would reach the close conclusion that SCAMP is more power-efficient than a normal CPU or GPU when performing image convolution operations. Notice that SCAMP utilises a less advanced 180 *nm* manufacturing technology and a calculation frequency of just 10 MHz in comparison to state-of-the-art CPU/GPUs. In addition, the Vision Processing Unit (VPU)[1] is a new type of processor built primarily to accelerate convolutional neural network inference while consuming less power (2 – 3 W) than the GPU (around 75 W). While the CPU, GPU, and VPU are all intended to do computations, SCAMP enables the effective integration of sensing, storage, and processing into a sensory chip with a general-purpose use. However, due to hardware resource constraints, the current SCAMP vision system can only execute shallow neural networks, which makes it still challenging to carry out practical neural network tasks as the CPU/GPU/VPU did.

[1] https://www.intel.com/content/www/us/en/products/details/processors/movidius-vpu.html

## 6.3 Future Work

We view these constraints as catalysts for research opportunities rather than as impediments to future progress. Regarding these constraints, we identify the following directions for further research as promising.

### 6.3.1 Unconventional Computing on the Sensor

With these above-mentioned limitations in mind, novel computing architecture and methods for on-sensor computing need investigating. For example, some advances have been made in unconventional computing like Neural Cellular Automata [84], Elementary Cellular Automata (Figure 1.14), Neuromorphic computing, Transformers [194], graph neural network [195], and reservoir computing [196]. In particular, the Cellular Automata-based reservoir computing [197] has made some headway in the sequential information process, which mimicks the mechanism of the recurrent neural network.

### 6.3.2 High-Performance Neural Network Quantisation and Deployment

Considering the issue of drop in accuracy described in Section 6.2.3, high-performance neural network quantisation is needed. The binarisation technique is utilised in this work. However, further network compression approaches as mentioned in Section 1.5.1 can be investigated for enhanced neural network compression performance. In addition, associated network deployment techniques should be proposed to close the performance gap between off-sensor simulation and on-sensor inference. Furthermore, specialised programming tools, notably for neural networks, are required to simply download a trained neural network model into the PPA without implementing it layer by layer.

### 6.3.3 Visual-servo System for Agile Robotics

This thesis has exhibited direct servo control with on-sensor visual inference using SCAMP-5d vision system in Section 3.6, demonstrating the capability to execute high-speed servo control with low latency and power consumption on a relatively modest experiment platform. In addition, the newly built SCAMP-7 vision system (Figure 6.1) by the University of Manchester enjoys more AREG computing resources with a smaller volume and weight. As a result, it has the potential to unleash a new avenue for future

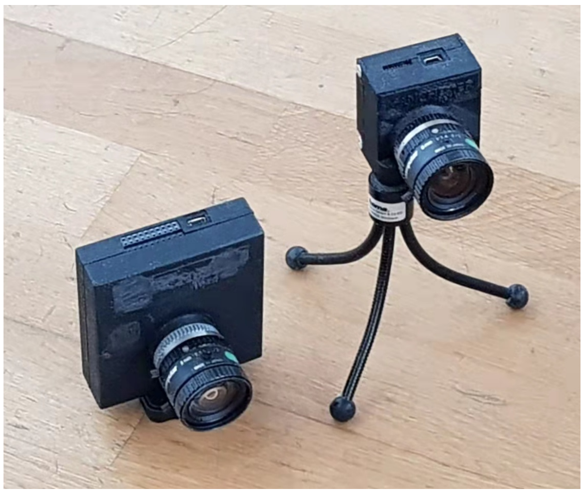


Figure 6.1: SCAMP 5 (Left) and SCAMP 7 (Right) vision system.

research into developing a visual-servo system based on the SCAMP vision system for agile exploration employing high-speed mobile robots, such as a fixed-wing aeroplanes.

### 6.3.4 Sensor Fusion

A variety of studies and applications have been proposed with a single gray-scale SCAMP PPA. However, some research, such as stereo or SLAM that usually needs the multiple sensors' fusion, are tricky for a single SCAMP PPA. Hence, the fusion of multiple SCAMP PPAs or PPA with other types of sensors can be a future direction that can further explore the on-sensor ability of the SCAMP PPA.

## 6.4 Conclusion

The purpose of this thesis was to investigate on-sensor visual signal processing for robotic applications and neural network inferences, with the goal of minimising the on-sensor vision system's latency and throughput. This was demonstrated by performing on-sensor feature extraction for robotics and on-sensor neural network tasks, which are typically performed by conventional machine vision systems using digital image processing. As a result, the thesis has placed a strong emphasis on parallel visual signal processing techniques and their efficient implementation.

Implementing robot vision and neural network inference systems is tough, even more so under severe power and weight limits, due to the amount of computer power required to make sense of the visual world. Conventional video cameras generate an enormous amount of raw data, which results in massive amounts of often unnecessary data being sent and processed through the visual pipeline. This gives an opportunity for unique vision sensor hardware that minimises this information flow by extracting features and other high-level data at the source. Our proposed methods in this thesis based on PPA devices enable processing to take place directly within the image sensor, extracting important data and outputting just high-level data rather than image frames. The PPA technique is defined by information extraction and data compression at the pixel level, which leads to an efficient and high-speed data transfer from the sensor device to the rest of the system.

In the near future, more advanced research outputs with a more optimised PPA is likely. Collaboration between chip engineers, image processing researchers, and roboticists are required to realise the vast potential of on-sensor processing technologies, beginning with low-level VLSI design and progressing through fundamental algorithms and practical applications to bring PPA out of the lab and into the real world. PPA can be viewed as a young, interdisciplinary, and fertile research platform for the study of analogue signal processing, fundamental parallel imaging algorithms, and innovative concurrent perception and processing for mobile/embedded systems.

APPENDIX A

# APPENDIX A

This parts presents useful tools for image processing algorithms development, algorithms, and robot-related application exploration and validation using the real/simulated SCAMP vision system.

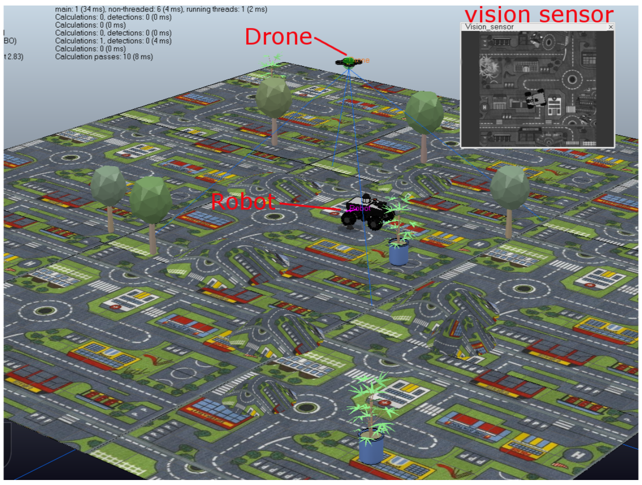
## A.1 Useful Tools for the SCAMP Vision System

As an emerging novel vision sensor, it is essential to have user-friendly developing and simulating tools both to researchers and engineers for idea validations.

### A.1.1 Development Platforms and Frameworks

There are comprehensive developing tools for this emerging on-sensor computing device to aid both researchers and engineers in developing their projects. Chen *et al.* [39] developed the guidance and documents, libraries (low and mid-level basic image processing algorithms), simulation and development environment configuration based on a computer (details can be seen from[1]).

[1]https://scamp.gitlab.io/scamp5d_doc/



```
main: 1 (34 ms), non-threaded: 6 (4 ms), running threads: 1 (2 ms)
Calculations: 0, detections: 0 (0 ms)
Calculations: 0 (0 ms)
Calculations: 0, detections: 0 (0 ms)
Calculations: 1, detections: 0 (4 ms)
Calculations: 0 (0 ms)
Calculation passes: 10 (8 ms)
```

Figure A.1: Example robot simulation environment. The virtual SCAMP camera sensor is 'mounted' under the drone facing the ground. Real-time image can be obtained from the sensor with a resolution of 256×256 which is set the same as that of the SCAMP vision system. Note that the current version of SCAMP-5d only supports gray-scale images; hence CNN inference on SCAMP only relies on gray-level features from the scene.

### A.1.2 Semi-Simulated Platform Development for the Full-Loop Robot Control

Based on the proposed neural network structure in the above-mentioned context, this section develops and demonstrates the integration of the SCAMP-5d vision system into the CoppeliaSim robot simulator, creating a semi-simulated environment. By configuring a camera in the simulator and setting up communication with the SCAMP Python host through remote API, sensor images from the simulator can be transferred to the SCAMP vision sensor, where on-sensor image processing such as CNN inference can be performed. SCAMP output is then fed back into CoppeliaSim. This proposed platform integration enables rapid prototyping validations of SCAMP algorithms for robotic systems. We demonstrate a rover localisation and tracking task using this proposed semi-simulated platform, with a CNN inference on SCAMP to command the motion of a robot. We made this platform available online: `https://github.com/yananliusdu/scamp5d_interface`.

The SCAMP vision system [39] is a smart camera device supporting in-sensor processing. The mixed-signal (analogue and digital), general-purpose processing circuits, integrated with image sensors in a pixel processor array (PPA), enable low-power consumption, and efficient parallel computing without external hardware. With these features, it

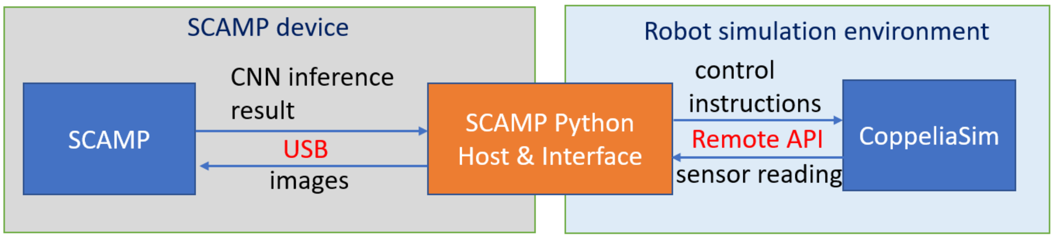


Figure A.2: Semi-simulated platform by integrating the SCAMP with CoppeliaSim Robot Simulator. This platform takes advantage of the SCAMP parallel computation ability and the rich simulation scenes in the simulator, where applications can be exploited and validated virtually.

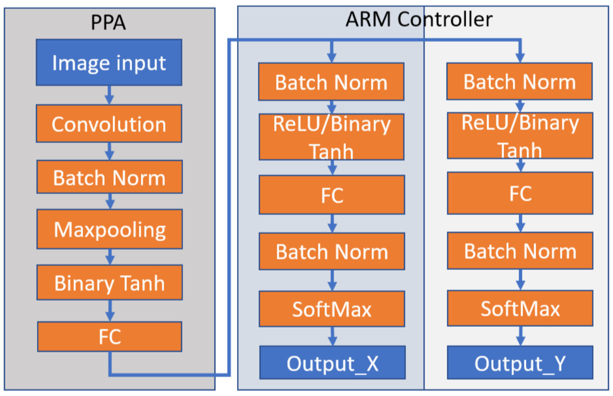


Figure A.3: The CNN architecture for localisation. A shared convolutional layer is used for object 2D localisation on the SCAMP, where networks for labels indicating $x$ and $y$ are using identical convolutional layers but different fully-connected layers.

is increasingly being integrated with robots for various applications [44, 45, 198, 199]. However, it is often time-consuming and difficult to prototype ideas using real robotic platforms. To improve experimental flexibility, we integrated a comprehensive robot simulator CoppeliaSim[200] with the SCAMP hardware system, to test and validate ideas rapidly (Figure A.1). CoppeliaSim is a robot environment simulator where each agent can be controlled via remote API [201]. Its simulated sensor readings can be transferred to other independent platforms written in Python, C/C++, or Matlab through several communication protocols. We developed a Python-based interface between CoppeliaSim and the SCAMP vision system. Based on the proposed semi-simulation platform, we implemented a convolutional neural network (CNN) [65, 175] on the SCAMP, processing the imported camera images for localisation purpose from the robot simulator where the camera is mounted under a drone.

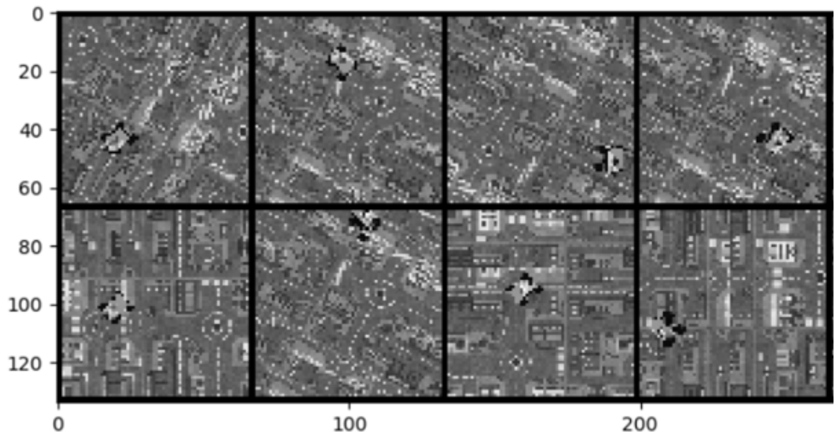


Figure A.4: Selected images for training. The inputs for SCAMP are gray-scale images with a low resolution of 64×64 considering the network architecture for parallel computation purpose, resulting in a challenging localisation task.

#### A.1.2.1 SCAMP Python Host and the Semi-Simulated Platform

The SCAMP vision system is connected to a computer via USB through scamp5d_interface library[2]. SCAMP Host is a GUI executed on the computer to interact with the vision system and visualise the data sent back from the device. This work develops a scamp_python_module for the Python GUI based on previous C/C++ host libraries, to simplify the connection of the SCAMP host to third party software. With this method, the host visualisation and remote API can be co-designed on the scamp_python_module. The remote API is supported by the CoppeliaSim[3].

We demonstrated an application of a CNN-based vision tracking task performed in our environment. In the semi-simulated platform, a real (hardware) SCAMP vision system collaborates with CoppeliaSim robotics simulation environment through the remote API. The environment setup and sensor image collection is performed in the simulator, while the SCAMP hardware is in charge of CNN inference with sensor images from the simulator and outputting useful information to the simulator.

In detail, as seen in Figure A.2, the synthetic picture generated by the visual sensor in the simulation is first delivered to the SCAMP Python Host & Interface, followed by image transmission to SCAMP hardware via USB. Following on-sensor computation, the rover's location information is relayed back to the simulation environment, instructing the drone's movement to track the rover. This establishes a complete closed-loop control

[2]https://scamp.gitlab.io/scamp5d_doc/
[3]https://www.coppeliarobotics.com/helpFiles/en/b0RemoteApiOverview.htm

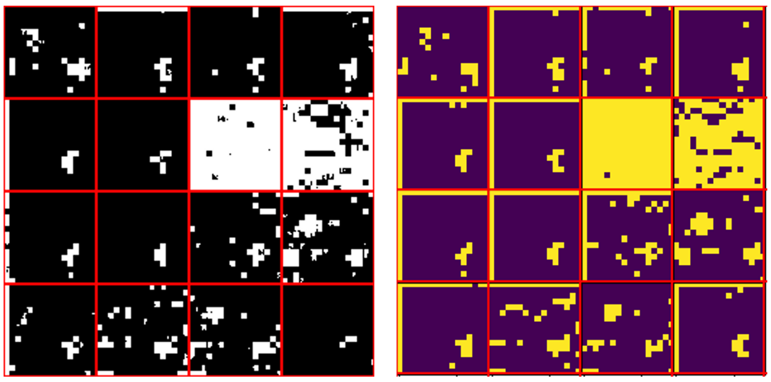


Figure A.5: Binary activations comparison after image convolution on SCAMP and using PyTorch simulation. White and yellow dots represent '1's while the dark area is '-1's. This shows the similarity of binary activations after the first convolutional layer, but with differences introduced due to non-idealities of analogue computing in hardware

between a real SCAMP hardware and a robot simulation environment. Note that the whole loop's running time depends on the computer performance for the simulator, the synthetic image generation & transmission, and the processing time on the SCAMP hardware.

#### A.1.2.2 Experiments on platform

The SCAMP vision system is suitable for mobile robot platforms due to its combination of high performance, low power consumption and light weight. In the experiments presented in this study, the SCAMP system is mounted on an aerial vehicle (drone), and used to localise a mobile ground vehicle (rover) moving in the 2D simulated environment. The location information is used to guide the drone to track the rover. We implemented the localisation task using a neural network.

The localisation training dataset is collected from the simulation environment, by placing the rover at a series of positions within the map, with different orientations under the view-field of a camera. An image is recorded once there is a change in the rover pose or camera pose. With this method, a dataset of 104,000 training images and 19,200 testing images was collected (Figure A.4). To simulate the vibration and tilting of a flying drone, random noise is introduced into the camera pose, and this can also be regarded as a type of data augmentation that benefits CNN training.

The localisation task is cast as a classification problem by splitting the $x$ and the $y$ axes into eight labels, giving 64 possible positions to localise the rover. During the CNN training, the training loss for backpropagation is the loss summation of $x$ and $y$ as

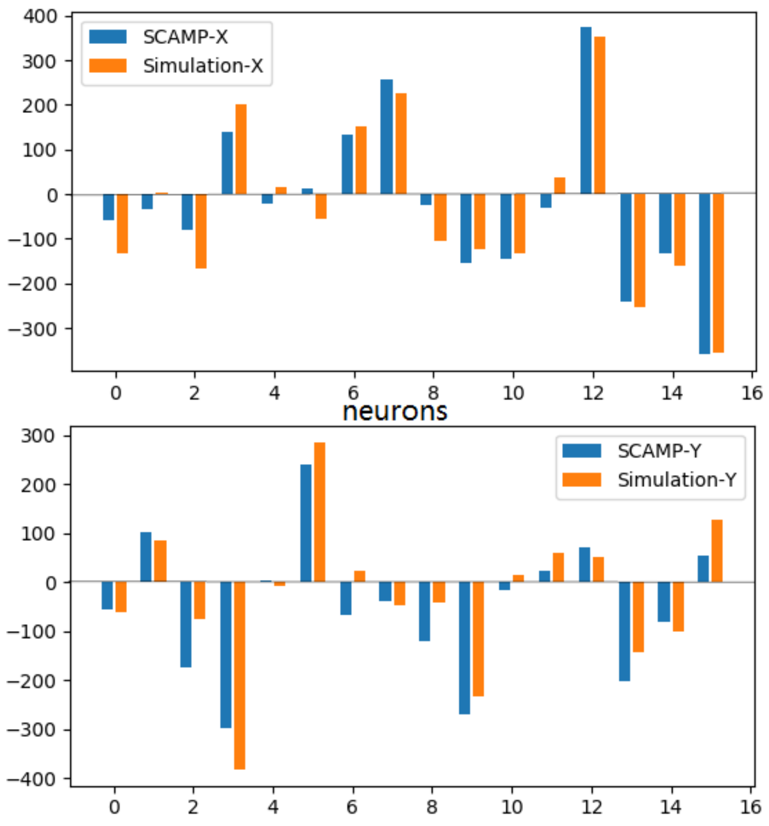


Figure A.6: Neuron activations after the first fully connected computed using SCAMP hardware and PyTorch simulation. Binary activations and bit counting strategies are used to mitigate against the inherent noise of the analogue processing on SCAMP, resulting in similar activation values in fully-simulated and in hardware implementations.

shown in Figure A.3. We trained a binarized CNN using batch normalisation and using both binary weights and activations to reduce the error caused by continuous analogue electronic current computing. The final testing accuracy for localisation is around 93%, which conservatively only counts the correct predictions of the CNN. In a practical situation, a close prediction to the ground truth should still allow tracking to proceed without significant difficulty. Figure A.7 visualises a sequence of 8 frames with CNN inference on SCAMP where the prediction possibility distribution can be seen plotted as light blue curves along $x$ and $y$ axes, the final prediction is obtained with two highest possibilities from two curves along the axes. The complete localisation and tracking are processed frame by frame, and the instructions to pilot the drone are generated using the PID control to minimise the distance between the drone position and the predicted rover position.

To demonstrate the CNN inference on SCAMP in terms of accuracy, Figure A.5 and

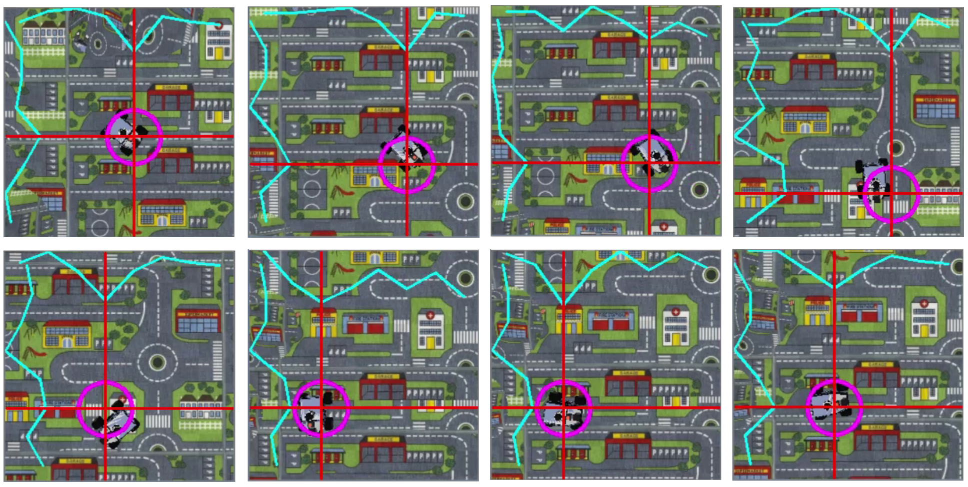


Figure A.7: Rover localisation result visualisation, showing SCAMP CNN inference results for several consecutive frames. The network outputs are plotted as light blue curves along $x$ and $y$ axes, with the largest values for each axis (red straight line) indicating the final 2D localisation prediction (pink circle). The full experimental video for rover localisation and tracking can be seen from https://youtu.be/semthdfXH5A.

Figure A.6 compare the binary activation layer and first fully-connected layer between PyTorch software implementation and SCAMP hardware implementation, which shows a high degree of similarity between PyTorch and SCAMP results, but with differences resulting from analogue signal processing on SCAMP. To further validate the performance of CNN localisation on SCAMP, a chaotic trajectory is pre-set in the simulator for the rover to move along. The drone trajectory is plotted with guidance from SCAMP CNN inference. Figure A.8 and A.9 show the comparison among the ground truth rover course, CNN on PyTorch guided drone course and CNN on SCAMP guided drone course. Table A.1 shows the performance of tracking by simulation and on-sensor computing. Again, the fully simulated results are similar, but not identical to the semi-simulated results using SCAMP hardware. These results indicate that the methodology based on training neural networks in software, using established frameworks such a PyTorch, and then implementing them on SCAMP analogue hardware, can produce useful results. However, software and hardware performances are not identical since the analogue hardware effects cannot always be accurately modelled. It is useful to be able to perform hardware-in-the-loop simulations combining the virtual environments and the SCAMP hardware, as presented in this study. We make our SCAMP Python host, CoppeliaSim model and its configuration available online: https://github.com/yananliusdu/scamp5d_interface.

In this work, we proposed a semi-simulated platform where a SCAMP hardware

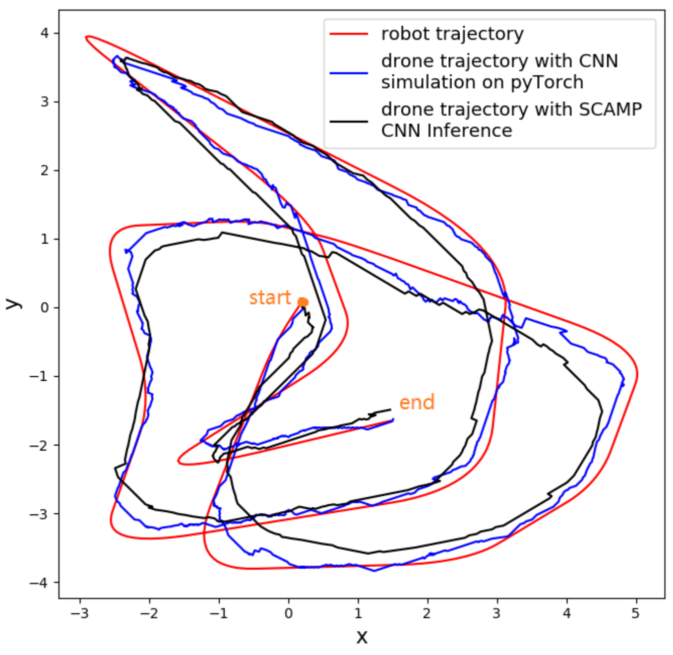


Figure A.8: Tracking trajectories with a drone. There are three paths: the pre-set rover trajectory as the groundtruth, drone tracking trajectory with CNN on PyTorch guidance, and drone tracking trajectory with guidance from SCAMP CNN inference.

Table A.1: Trajectory tracking error.

| trajectory | average localisation error | max localisation error |
|---|---|---|
| Drone tracking trajectory with pyTorch | 0.11 | 0.51 |
| Drone tracking trajectory with PPA | 0.21 | 0.77 |

interacts with the robot simulator via remote API for a rapid prototype validation. The SCAMP CNN inference results with the simulated sensor readings can instruct the motion of an agent in the proposed platform. Further applications related to the SCAMP and robots integration can be easily explored based on the developed platform.

We are using simulated data on the real PPA hardware for repeatability, allowing us to have a baseline and establish a process for evaluating and comparing against upcoming progress on future PPA architectures. As new PPA hardware is developed, studies will be able to baseline against our work here and benefit from the code and data we will release after publication.

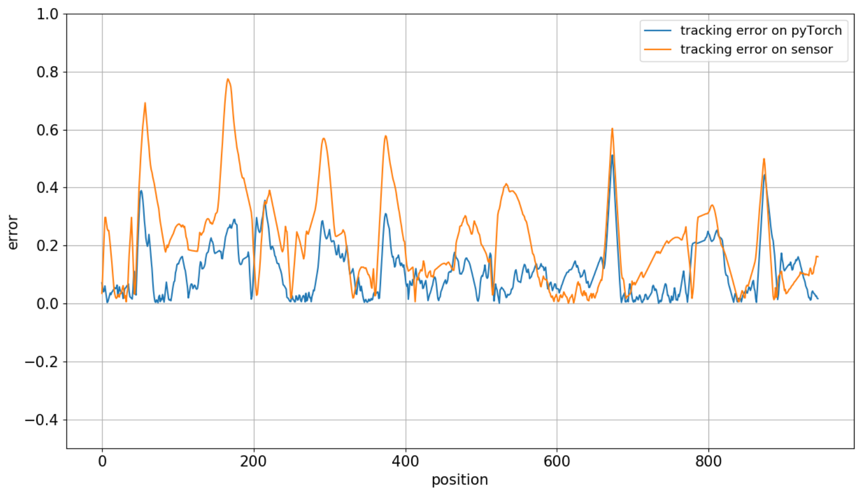


Figure A.9: Tracking errors with pyTorch simulation and on sensor.

### A.1.3 Fully-simulated Development Platform

After the semi-simulated platform, we ([202], Figure A.10) further developed a fully-simulated environment integrating the SCAMP server and Coppeliasim robot simulator, where both a simulated SCAMP vision system and a robot simulator (Coppeliasim) can communicate bidirectionally through remote API. Within the platform, a monocular vision-based mobile robot obstacle avoidance and navigation application are demonstrated, providing an example of robot-related prototype.

### A.1.4 Kernel Filter Compiler

Performing image convolution is necessary, even for middle-level image processing or convolutional neural network. Other than the convolution with binary/ternary weights mentioned above, approximated kernel filters for convolution is proposed by [72] with automatic code generation. In their work, each full-precision coefficient in the kernel filters is approximated by the combination of multiple addition/subtractions and divisions (Equ. A.1).

$$\alpha \approx \sum_{i=0}^{n} a_i/2^i, \qquad a_i \in \{-1,0,1\} \tag{A.1}$$

where, $n$ is the depth of the approximation. Their work is an effective way to approximate the full-precision kernel filters and automatically generate codes for the unique hardware, which is friendly to users. However, there are still some shortcomings compared to

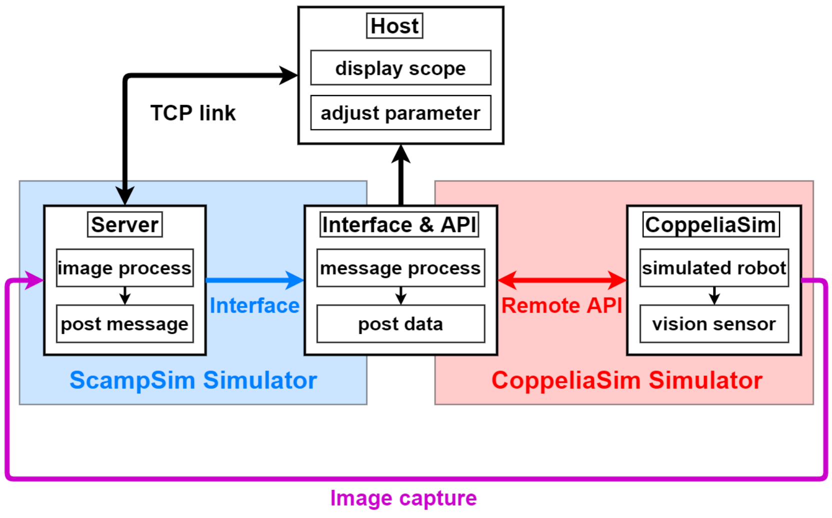


Figure A.10: The framework of the fully-simulated system. Once the Server is started, the Host works via TCP link. Then the CoppeliaSim is switched on by the remote API. After that, a closed loop of image processing from ScampSim, data transmission through interface/API, robot simulation and sensor reading from CoppeliaSim is set up, coming along with visualisation and parameter tuning on the Host. (Figure from [202])

binary/ternary kernel filters, including the introduced error by approximation, difficult to perform parallel multiple convolution with many filters. This results in inaccurate and inefficient CNN inference results when the neural network is deeper along with more layers and kernel filters. With similar coefficient approximation strategy, Stow at al. [73, 203] automatically generated codes for PPA which outperforms earlier AUKE in terms of simultaneous kernel optimisation, and generated more efficient codes. In addition, Stow *et al*. use the 4×4 'Super Pixel' method in their work [204], which was inspired by Bose's work [63] and their earlier compilation method to automates generation of SIMD code for convolutions using super-pixels.

### A.1.5 SCAMP-5 Registers' Setup and Instructions

In addition to the hardware architecture described in Figure 1.2 and the development process in Figure 1.4, the more detailed instruction sets and register resources for the SCAMP-5 can be seen from https://scamp.gitlab.io/scamp5d_doc/_p_a_g_e__d_e_v_i_c_e__a_p_i__c_a_t_e_g_o_r_y.html.

## A.2 Weights Estimation Algorithm

This section describes the weights estimation algorithms to update the node weights during the localisation process for Chapter 2.

**Algorithm 7:** Weights estimation algorithm [7].

```
if (database_exists) then
    load_database();
else
    mapping();
end
$t \leftarrow 0$;
initialise $\boldsymbol{X}_t$;
while input_frame do
    if (motion_model()) then
        for all the weights $i = 1, ..., N$ ;
        $x_t^{[i]} = motion\_model(x_{t-1}^{[i]})$ ;
        $\mu_t^{[i]} = likelihood(z_t | x_t^{[i]})$;
        $\boldsymbol{X}_t = \boldsymbol{X}_{t-1} + (x_t, \mu_t)$;
        if (t => sequence_length) then
            $prediction = max\_weight(\boldsymbol{X}_t)$ ;
        end
        if (t % sequence_length == 0) then
            $w_t^{[i]} = w_t^{[i]} - \gamma max(\boldsymbol{W}_t)$;
            $\boldsymbol{X}_t = \boldsymbol{X}_t + (\boldsymbol{W}_t)$;
        end
        $t \leftarrow t + 1$;
    end
end
```

## A.3 Motion Parallax Extraction

This section demonstrates the motion parallax extraction algorithms used to determine the maximum width contained inside an input binary motion parallax image.

**Algorithm 8:** Motion Parallax Extraction [67]

```
1  $A \leftarrow$ current input image.
2  $B \leftarrow$ last input image.
3  $t \leftarrow$ a threshold.
4  $R_w \leftarrow$ receptive field window Boolean image.
5  $init$:
6  $R_a \leftarrow |A - B| > t$.
7  $R_b \leftarrow R_a \wedge R_w$.
8  $steps \leftarrow 0$.
9  $loop$:
10 if $GlobalOr(R_b)$ then
11     $R_c \leftarrow \neg R_b$.
12     if $steps$ is odd then
13         $R_d \leftarrow North(R_c) \vee East(R_c)$.
14     else
15         $R_d \leftarrow South(R_c) \vee West(R_c)$.
16     if $R_d = 1$ then
17         $R_b \leftarrow 0$.
18     $steps \leftarrow steps + 1$
19     goto $loop$.
20 return $steps$
```

## A.4 Outdoor Demonstration of Mapping and Localisation

We took advantage of SCAMP-5d vision system in collecting both HDR and non-HDR images as datasets[4] around College Green (400 meters) to validate our proposed localisation methods. The mapping procedure is conducted by mounting the SCAMP vision system on the bike facing forward and then walking around the park for a circle. Notice that each image collected on the map is associated with a GPS location. Figure A.11

[4] https://uob-my.sharepoint.com/:f:/g/personal/yl17692_bristol_ac_uk/Eo9iA7LDLQNPgfRd6696MK4B4R1Ez05_o-v2iZEiSUw0bA?e=yS3nmQ

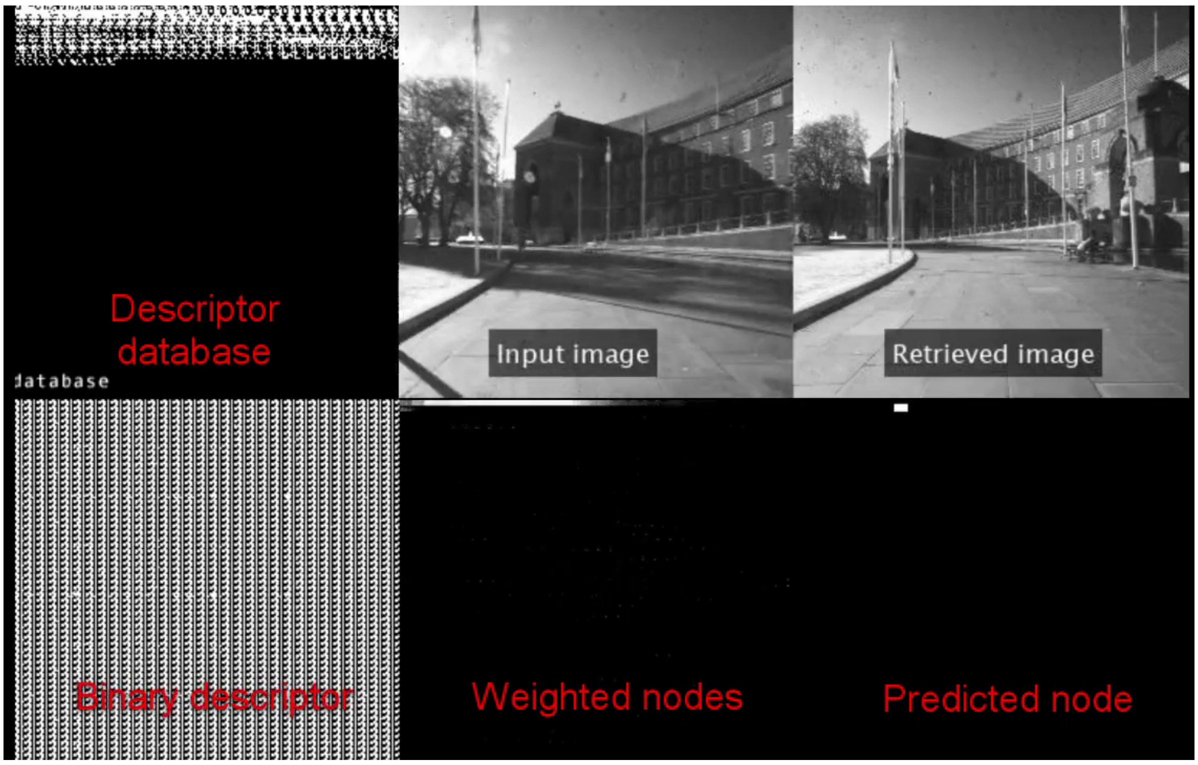


Figure A.11: Demonstration of real-time on-sensor localisation outdoors. The binary descriptor (bottom left) is generated by performing local binary pattern on the input image (top middle).

demonstrates the real-time outdoor test on the sensor in the College Green of Bristol, where the algorithms find the most likely image from the database by comparing image resemblance and applying the motion model.

## A.5 Rethinking a PhD during the COVID-19

Unexpectedly, two-thirds of my PhD study was conducted during the unprecedented coronavirus shutdown or restrictions with a transition from a normal PhD research to an unusual one with COVID-19. Nevertheless, virtual meeting platforms such as Skype and Zoom enabled me to continue to hold remote collaboration with my supervisors and other scholars. Furthermore, I appreciate that the University's procedures assisted us in safely navigating through this trying times with access to the facilities and CDT funding which enabled me to work from home. Thus, I was able to conduct experiments from home or at the office where possible.

As a PhD student who is supposed to carry out research in the area of robotics and autonomous systems, I changed my research direction to some degree, from the initial robotics-based research with experiments performed in the laboratory to on-sensor neural network inference with experiments from home or in the office due to the COVID-19. As demonstrated in Chapter 2, our primary focus is on machine vision and robotics until the coronavirus breakout in early 2020. Facing a laboratory lockdown without sufficient experimental facilities, I had to switch to a feasible research direction that requires minimal laboratory resources. Hence, I started to study embedded neural networks on the sensor as seen from Chapters 3, 4, and 5. After that, a semi-simulated robot-SCAMP platform and a fully simulated environment (Chapter 6) were developed to test our proposed algorithms and explore new line of research as well. With these schemes, I managed to obtain useful data which I successfully analysed for my PhD thesis.

# Bibliography

[1] Eye-RIS, "Eye-ris v1.3 hardware description," Accessed: 15-07-2009. http://users.itk.ppke.hu/~horan/EyeRis_documentation/Eye-RIS%20v1.3%20Hardware%20Description.pdf.

[2] C. Greatwood, L. Bose, T. Richardson, W. Mayol-Cuevas, R. Clarke, J. Chen, S. J. Carey, and P. Dudek, "Towards drone racing with a pixel processor array," in *11th International Micro Air Vehicles, Conferences and Competitions*, pp. 73–79, 2019.

[3] A. J. Lilienthal and T. Duckett, "Experimental analysis of smelling braitenberg vehicles," in *IEEE international conference on advanced robotics (ICAR 2003), Coimbra, Portugal, June 30-July 3, 2003*, vol. 1, pp. 375–380, Coimbra, University, 2003.

[4] S.-C. Liu and T. Delbruck, "Neuromorphic sensory systems," *Current opinion in neurobiology*, vol. 20, no. 3, pp. 288–295, 2010.

[5] Siliconretina, "User guides for DVS128 and DVS128_PAER Dynamic Vision Sensors, http://siliconretina.ini.uzh.ch/wiki/doku.php?id=userguide."

[6] F. Galluppi, C. Denk, M. C. Meiner, T. C. Stewart, L. a. Plana, C. Eliasmith, S. Furber, and J. Conradt, "Event-based neural computing on an autonomous mobile platform," *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2862–2867, 2014.

[7] H. Castillo-Elizalde, Y. Liu, L. Bose, and W. Mayol-Cuevas, "Weighted node mapping and localisation on a pixel processor array," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021.

[8] Á. Zarándy, *Focal-plane sensor-processor chips*.
Springer Science & Business Media, 2011.

[9] Microelectronics Design Lab, “SCAMP Vision Sensor,” 2018.

[10] S. J. Carey, A. Lopich, D. R. Barr, B. Wang, and P. Dudek, “A 100,000 fps vision sensor with embedded 535gops/w 256× 256 simd processor array,” in *2013 Symposium on VLSI Circuits*, pp. C182–C183, IEEE, 2013.

[11] P. Dudek and P. J. Hicks, “A cmos general-purpose sampled-data analog processing element,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 47, no. 5, pp. 467–473, 2000.

[12] P. Dudek, T. Richardson, L. Bose, S. Carey, J. Chen, C. Greatwood, Y. Liu, and W. Mayol-Cuevas, “Pixel processor array for sensor-level computer vision on agile robots,” *Science Robotics 2021, under review*.

[13] Y. Liu, T. Hu, H. Ni, C. Zhang, B. Lin, and E. Judd, “Design of a pc-based open industrial robot control system integrated with real-time machine vision,” in *2018 WRC Symposium on Advanced Robotics and Automation (WRC SARA)*, pp. 1–6, IEEE, 2018.

[14] S. Anubha Pearline, V. Sathiesh Kumar, and S. Harini, “A study on plant recognition using conventional image processing and deep learning approaches,” *Journal of Intelligent & Fuzzy Systems*, vol. 36, no. 3, pp. 1997–2004, 2019.

[15] F. Zhou and Y. Chai, “Near-sensor and in-sensor computing,” *Nature Electronics*, vol. 3, no. 11, pp. 664–671, 2020.

[16] K. Guo, S. Zeng, J. Yu, Y. Wang, and H. Yang, “A survey of fpga-based neural network accelerator,” *arXiv preprint arXiv:1712.08934*, 2017.

[17] Y. Chai, “In-sensor computing for machine vision,” 2020.

[18] D. H. Hubel, *Eye, brain, and vision.*
Scientific American Library/Scientific American Books, 1995.

[19] L. Mennel, J. Symonowicz, S. Wachter, D. K. Polyushkin, A. J. Molina-Mendoza, and T. Mueller, “Ultrafast machine vision with 2d material neural network image sensors,” *Nature*, vol. 579, no. 7797, pp. 62–66, 2020.

[20] R. Song, K. Huang, Z. Wang, and H. Shen, “An ultra fast low power convolutional neural network image sensor with pixel-level computing,” *arXiv preprint arXiv:2101.03308*, 2021.

[21] inivation, “Dynamic vision platform,” 2021.
https://inivation.com/dvp/.

[22] T.-Y. Wang, J.-L. Meng, Q.-X. Li, Z.-Y. He, H. Zhu, L. Ji, Q.-Q. Sun, L. Chen, and D. W. Zhang, “Reconfigurable optoelectronic memristor for in-sensor computing applications,” *Nano Energy*, vol. 89, p. 106291, 2021.

[23] Y. Wang, Y. Gong, L. Yang, Z. Xiong, Z. Lv, X. Xing, Y. Zhou, B. Zhang, C. Su, Q. Liao, *et al.*, “Mxene-zno memristor for multimodal in-sensor computing,” *Advanced Functional Materials*, vol. 31, no. 21, p. 2100144, 2021.

[24] L. Sun, Z. Wang, J. Jiang, Y. Kim, B. Joo, S. Zheng, S. Lee, W. J. Yu, B.-S. Kong, and H. Yang, “In-sensor reservoir computing for language learning via two-dimensional memristors,” *Science advances*, vol. 7, no. 20, p. eabg1455, 2021.

[25] Sony, “Intelligent vision sensors with ai processing functionality,” Accessed: 14-05-2020.
https://www.sony.com/en/SonyInfo/News/Press/202005/20-037E/.

[26] Aistorm, “Ai in sensor,” Accessed: 22-11-2021.
https://aistorm.ai/.

[27] P. Yao, H. Wu, B. Gao, J. Tang, Q. Zhang, W. Zhang, J. J. Yang, and H. Qian, “Fully hardware-implemented memristor convolutional neural network,” *Nature*, vol. 577, no. 7792, pp. 641–646, 2020.

[28] A. Thomas, “Memristor-based neural networks,” *Journal of Physics D: Applied Physics*, vol. 46, no. 9, p. 093001, 2013.

[29] I. A. Lungu, S.-C. Liu, and T. Delbruck, “Fast event-driven incremental learning of hand symbols,” in *2019 IEEE International Conference on Artificial Intelligence Circuits and Systems (AICAS)*, pp. 25–28, IEEE, 2019.

[30] I.-A. Lungu, F. Corradi, and T. Delbrück, “Live demonstration: Convolutional neural network driven by dynamic vision sensor playing roshambo,” in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–1, IEEE, 2017.

[31] A. Linares-Barranco, A. Rios-Navarro, R. Tapiador-Morales, and T. Delbruck, “Dynamic vision sensor integration on fpga-based cnn accelerators for high-speed visual classification,” *arXiv preprint arXiv:1905.07419*, 2019.

[32] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. Davison, J. Conradt, K. Daniilidis, *et al.*, "Event-based vision: A survey," *arXiv preprint arXiv:1904.08405*, 2019.

[33] P. Dudek, *A programmable focal-plane analogue processor array*. PhD thesis, 2000.

[34] S. Carey and P. Dudek, "Vision chip with high accuracy analog s 2 i cells," in *2014 14th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA)*, pp. 1–2, IEEE, 2014.

[35] S. J. Carey, D. R. Barr, B. Wang, A. Lopich, and P. Dudek, "Mixed signal simd processor array vision chip for real-time image processing," *Analog Integrated Circuits and Signal Processing*, vol. 77, no. 3, pp. 385–399, 2013.

[36] A. Lopich and P. Dudek, "A general-purpose vision processor with 160× 80 pixel-parallel simd processor array," in *Proceedings of the IEEE 2013 Custom Integrated Circuits Conference*, pp. 1–4, IEEE, 2013.

[37] J. N. Martel and P. Dudek, "Vision chips with in-pixel processors for high-performance low-power embedded vision systems," in *ASR-MOV Workshop, CGO*, vol. 6, p. 14, 2016.

[38] P. Dudek and P. J. Hicks, "An simd array of analogue microprocessors for early vision," in *Proc. Conf. Postgraduate Research in Electronics, Photonics and Related Fields (PREP'99)*, pp. 359–362, 1999.

[39] J. Chen, S. J. Carey, and P. Dudek, "Scamp5d vision system and development framework," in *Proceedings of the 12th International Conference on Distributed Smart Cameras*, pp. 1–2, 2018.

[40] P. Dudek, "Scamp-5: Vision sensor with pixel parallel simd processor array." https://youtu.be/D3VcmkQiPR4, 2019.

[41] J. Neuffer and T. Younkin, "Interim report for the decadal plan for semiconductors," pp. 1–21, Semiconductor research corporation and semiconductor industry association, 2020.

[42] C. Greatwood, L. Bose, T. Richardson, W. Mayol-Cuevas, J. Chen, S. J. Carey, and P. Dudek, "Tracking control of a uav with a parallel visual processor," in *2017*

*IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 4248–4254, IEEE, 2017.

[43] C. Greatwood, L. Bose, T. Richardson, W. Mayol-Cuevas, J. Chen, S. J. Carey, and P. Dudek, "Perspective correcting visual odometry for agile mavs using a pixel processor array," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 987–994, IEEE, 2018.

[44] A. McConville, L. Bose, R. Clarke, W. Mayol-Cuevas, J. Chen, C. Greatwood, S. Carey, P. Dudek, and T. Richardson, "Visual odometry using pixel processor arrays for unmanned aerial systems in gps denied environments," *Frontiers in Robotics and AI*, vol. 7, 2020.

[45] Y. Liu, L. Bose, C. Greatwood, J. Chen, R. Fan, T. Richardson, S. J. Carey, P. Dudek, and W. Mayol-Cuevas, "Agile reactive navigation for a non-holonomic mobile robot using a pixel processor array," *IET Image Processing*, pp. 1–10, 2021.

[46] S. J. Carey, D. R. Barr, and P. Dudek, "Low power high-performance smart camera system based on scamp vision sensor," *Journal of Systems Architecture*, vol. 59, no. 10, pp. 889–899, 2013.

[47] B. Wang and P. Dudek, "Amber: Adapting multi-resolution background extractor," in *2013 IEEE International Conference on Image Processing*, pp. 3417–3421, IEEE, 2013.

[48] B. Wang and P. Dudek, "A fast self-tuning background subtraction algorithm," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 395–398, 2014.

[49] P. Dudek, "Adaptive sensing and image processing with a general-purpose pixel-parallel sensor/processor array integrated circuit," in *2006 International Workshop on Computer Architecture for Machine Perception and Sensing*, pp. 1–6, IEEE, 2006.

[50] P. Dudek and D. L. Vilariño, "A cellular active contours algorithm based on region evolution," in *2006 10th International Workshop on Cellular Neural Networks and Their Applications*, pp. 1–6, IEEE, 2006.

[51] C. Alonso-Montes, D. Vilarino, and M. Penedo, “Cnn-based automatic retinal vascular tree extraction,” in *2005 9th International Workshop on Cellular Neural Networks and Their Applications*, pp. 61–64, IEEE, 2005.

[52] B. Wang, P. Mroszczyk, and P. Dudek, “A new method for fast skeletonization of binary images on cellular processor arrays,” in *2014 14th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA)*, pp. 1–2, IEEE, 2014.

[53] P. Mroszczyk and P. Dudek, “Trigger-wave collision detecting asynchronous cellular logic array for fast image skeletonization,” in *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 2653–2656, IEEE, 2012.

[54] S. Razmjooei and P. Dudek, “Approximating euclidean distance transform with simple operations in cellular processor arrays,” in *2010 12th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA 2010)*, pp. 1–5, IEEE, 2010.

[55] J. N. Martel, L. K. Müller, S. J. Carey, and P. Dudek, “Parallel hdr tone mapping and auto-focus on a cellular processor array vision chip,” in *2016 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1430–1433, IEEE, 2016.

[56] J. N. Martel, *Unconventional processing with unconventional visual sensing: Parallel, distributed and event based vision algorithms & systems*.
PhD thesis, ETH Zurich, 2019.

[57] J. N. Martel, L. K. Mueller, S. J. Carey, P. Dudek, and G. Wetzstein, “Neural sensors: Learning pixel exposures for hdr imaging and video compressive sensing with programmable sensors,” *IEEE transactions on pattern analysis and machine intelligence*, vol. 42, no. 7, pp. 1642–1653, 2020.

[58] J. Chen, S. J. Carey, and P. Dudek, “Feature extraction using a portable vision system,” in *IEEE/RSJ Int. Conf. Intell. Robots Syst., Workshop Vis.-based Agile Auton. Navigation UAVs*, 2017.

[59] L. Bose, J. Chen, S. J. Carey, P. Dudek, and W. Mayol-Cuevas, “Visual odometry for pixel processor arrays,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 4604–4612, 2017.

[60] R. Murai, S. Saeedi, and P. H. Kelly, "Bit-vo: Visual odometry at 300 fps using binary features from the focal plane," *arXiv preprint arXiv:2004.11186*, 2020.

[61] S. J. Carey, D. R. Barr, B. Wang, A. Lopich, and P. Dudek, "Locating high speed multiple objects using a scamp-5 vision-chip," in *2012 13th International Workshop on Cellular Nanoscale Networks and their Applications*, pp. 1–2, IEEE, 2012.

[62] D. R. Barr, S. J. Carey, and P. Dudek, "Low power multiple object tracking and counting using a scamp cellular processor array," in *2012 13th International Workshop on Cellular Nanoscale Networks and their Applications*, pp. 1–2, IEEE, 2012.

[63] L. Bose, J. Chen, S. J. Carey, P. Dudek, and W. Mayol-Cuevas, "A camera that cnns: Towards embedded neural networks on pixel processor arrays," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1335–1344, 2019.

[64] L. Bose, P. Dudek, J. Chen, S. J. Carey, and W. W. Mayol-Cuevas, "Fully embedding fast convolutional networks on pixel processor arrays," in *European Conference on Computer Vision*, pp. 488–503, Springer, 2020.

[65] Y. Liu, L. Bose, J. Chen, S. J. Carey, P. Dudek, and W. Mayol-Cuevas, "High-speed light-weight cnn inference via strided convolutions on a pixel processor array," in *The 31st British Machine Vision Conference (BMVC)*, 2020.

[66] Y. Liu, L. Bose, Y. Lu, P. Dudek, and W. Mayol-Cuevas, "On-sensor binarized fully convolutional neural network with a pixel processor array," *arXiv preprint arXiv:2202.00836*, 2022.

[67] J. Chen, Y. Liu, S. J. Carey, and P. Dudek, "Proximity estimation using vision features computed on sensor," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2689–2695, IEEE, 2020.

[68] M. Z. Wong, B. Guillard, R. Murai, S. Saeedi, and P. H. Kelly, "Analognet: Convolutional neural network inference on analog focal plane sensor processors," *arXiv preprint arXiv:2006.01765*, 2020.

[69] J. N. Martel, L. K. Müller, S. J. Carey, J. Müller, Y. Sandamirskaya, and P. Dudek, "Real-time depth from focus on a programmable focal plane processor," *IEEE*

*Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 3, pp. 925–934, 2017.

[70] J. N. Martel, L. K. Müller, S. J. Carey, and P. Dudek, "High-speed depth from focus on a programmable vision chip using a focus tunable lens," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, IEEE, 2017.

[71] J. N. Martel, L. K. Müller, S. J. Carey, J. Müller, Y. Sandamirskaya, and P. Dudek, "Live demonstration: Depth from focus on a focal plane processor using a focus tunable liquid lens," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–1, IEEE, 2017.

[72] T. Debrunner, S. Saeedi, and P. H. Kelly, "Auke: Automatic kernel code generation for an analogue simd focal-plane sensor-processor array," *ACM Transactions on Architecture and Code Optimization (TACO)*, vol. 15, no. 4, pp. 1–26, 2019.

[73] E. Stow, R. Murai, S. Saeedi, and P. H. Kelly, "Cain: Automatic code generation for simultaneous convolutional kernels on focal-plane sensor-processors," *arXiv preprint arXiv:2101.08715*, 2021.

[74] B. Wang and P. Dudek, "Coarse grain mapping method for image processing on fine grain cellular processor arrays," in *2012 13th International Workshop on Cellular Nanoscale Networks and their Applications*, pp. 1–6, IEEE, 2012.

[75] J. N. Martel, Y. Sandamirskaya, and P. Dudek, "A demonstration of tracking using dynamic neural fields on a programmable vision chip," in *Proceedings of the 10th International Conference on Distributed Smart Camera*, pp. 212–213, 2016.

[76] P. Dudek and S. Carey, "General-purpose 128/spl times/128 simd processor array with integrated image sensor," *Electronics Letters*, vol. 42, no. 12, pp. 678–679, 2006.

[77] J. N. Martel, L. K. Müller, S. J. Carey, and P. Dudek, "A real-time high dynamic range vision system with tone mapping for automotive applications," in *CNNA 2016; 15th International Workshop on Cellular Nanoscale Networks and their Applications*, pp. 1–2, VDE, 2016.

[78] P. Dudek, M. Hülse, and D. R. Barr, “Cellular automata and non-static image processing for embodied robot systems on a massively parallel processor array,” in *Automata-2008: Theory and Applications of Cellular Automata*, pp. 504–510, Luniver Press, 2008.

[79] A. J. Davison, “Futuremapping: The computational structure of spatial ai systems,” *arXiv preprint arXiv:1803.11288*, 2018.

[80] T. Debrunner, S. Saeedi, L. Bose, A. J. Davison, and P. H. Kelly, “Camera tracking on focal-plane sensor-processor arrays,” in *Proceedings of the Workshop on Programmability and Architectures for Heterogeneous Multicores (MULTIPROG), Vancouver, BC, Canada*, vol. 15, 2019.

[81] G. Klein and D. Murray, “Parallel tracking and mapping for small ar workspaces,” in *2007 6th IEEE and ACM international symposium on mixed and augmented reality*, pp. 225–234, IEEE, 2007.

[82] S. Wolfram, “Cellular automata as models of complexity,” *Nature*, vol. 311, no. 5985, pp. 419–424, 1984.

[83] P. L. Rosin, “Training cellular automata for image processing,” *IEEE transactions on image processing*, vol. 15, no. 7, pp. 2076–2087, 2006.

[84] A. Mordvintsev, E. Randazzo, E. Niklasson, and M. Levin, “Growing neural cellular automata,” *Distill*, vol. 5, no. 2, p. e23, 2020.

[85] Y. Cheng, D. Wang, P. Zhou, and T. Zhang, “A survey of model compression and acceleration for deep neural networks,” *arXiv preprint arXiv:1710.09282*, 2017.

[86] S. Gupta, A. Agrawal, K. Gopalakrishnan, and P. Narayanan, “Deep learning with limited numerical precision,” in *International conference on machine learning*, pp. 1737–1746, PMLR, 2015.

[87] V. Vanhoucke, A. Senior, and M. Z. Mao, “Improving the speed of neural networks on cpus,” 2011.

[88] F. Li, B. Zhang, and B. Liu, “Ternary weight networks,” *arXiv preprint arXiv:1605.04711*, 2016.

[89] M. Courbariaux, Y. Bengio, and J.-P. David, "Binaryconnect: Training deep neural networks with binary weights during propagations," in *Advances in neural information processing systems*, pp. 3123–3131, 2015.

[90] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to+ 1 or- 1. arxiv 2016," *arXiv preprint arXiv:1602.02830*.

[91] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European Conference on Computer Vision*, pp. 525–542, Springer, 2016.

[92] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[93] C. Zhao, B. Ni, J. Zhang, Q. Zhao, W. Zhang, and Q. Tian, "Variational convolutional neural network pruning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[94] W. Wen, C. Wu, Y. Wang, Y. Chen, and H. Li, "Learning structured sparsity in deep neural networks," *Advances in neural information processing systems*, vol. 29, pp. 2074–2082, 2016.

[95] B. Chen, Z. Yang, and Z. Yang, "An algorithm for low-rank matrix factorization and its applications," *Neurocomputing*, vol. 275, pp. 1012–1020, 2018.

[96] J. Gou, B. Yu, S. J. Maybank, and D. Tao, "Knowledge distillation: A survey," *International Journal of Computer Vision*, vol. 129, no. 6, pp. 1789–1819, 2021.

[97] F. Tung and G. Mori, "Similarity-preserving knowledge distillation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1365–1374, 2019.

[98] J. A. Tobaruela and A. O. Rodríguez, *Reactive navigation in extremely dense and highly intricate environments*, vol. 12.
2017.

[99] H. Li and S. X. Yang, "A behavior-based mobile robot with a visual landmark-recognition system," *IEEE/ASME Transactions on Mechatronics*, vol. 8, no. 3, pp. 390–400, 2003.

[100] M. Mujahed, D. Fischer, and B. Mertsching, “Safe Gap based (SG) reactive navigation for mobile robots,” *2013 European Conference on Mobile Robots, ECMR 2013 - Conference Proceedings*, pp. 325–330, 2013.

[101] R. A. Brooks, “Intelligence without representation,” *Artificial Intelligence*, vol. 47, no. 1-3, pp. 139–159, 1991.

[102] C. Pêtrès, M. A. Romero-Ramirez, F. Plumet, and B. Alessandrini, “Modeling and reactive navigation of an autonomous sailboat,” *IEEE International Conference on Intelligent Robots and Systems*, pp. 3571–3576, 2011.

[103] B. Penin, P. R. Giordano, F. Chaumette, B. Penin, P. R. Giordano, F. Chaumette, and V.-b. R. Planning, “Vision-Based Reactive Planning for Aggressive Target Tracking while Avoiding Collisions and Occlusions To cite this version : Vision-Based Reactive Planning for Aggressive Target Tracking while Avoiding Collisions and Occlusions,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3725–3732, 2018.

[104] P. Lichtsteiner, C. Posch, T. Delbruck, and S. Member, “A 128 128 120 dB 15 us Latency Asynchronous Temporal Contrast Vision Sensor,” *Work*, vol. 43, no. 2, pp. 566–576, 2008.

[105] L. Everding and J. Conradt, “Low-latency line tracking using event-based dynamic vision sensors,” *Frontiers in Neurorobotics*, vol. 12, no. FEB, pp. 1–13, 2018.

[106] E. Mueggler, N. Baumli, F. Fontana, and D. Scaramuzza, “Towards evasive maneuvers with quadrotors using dynamic vision sensors,” *2015 European Conference on Mobile Robots, ECMR 2015 - Proceedings*, 2015.

[107] A. I. Maqueda, A. Loquercio, G. Gallego, N. Garcia, and D. Scaramuzza, “Event-based Vision meets Deep Learning on Steering Prediction for Self-driving Cars,” no. Dl, pp. 5419–5427, 2018.

[108] L. Bose, J. Chen, S. J. Carey, P. Dudek, and W. Mayol-Cuevas, “Visual Odometry for Pixel Processor Arrays,” *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2017-Octob, pp. 4614–4622, 2017.

[109] Y. Liu, L. Bose, J. Chen, R. Fan, P. Dudek, and W. Mayol-Cuevas, “On-sensor cnn parallel computing with a pixel processor array,” *IEEE Transaction on Computational Imaging, manuscript*, 2021.

[110] L. Bose, J. Chen, S. J. Carey, P. Dudek, and W. Mayol-Cuevas, “A camera that cnns: Towards embedded neural networks on pixel processor arrays,” in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2019.

[111] Y. Liu, J. Chen, L. Bose, P. Dudek, and W. Mayol-Cuevas, “Direct servo control from in-sensor cnn inference with a pixel processor array,” in *2021 IEEE International Conference on Robotics and Automation (ICRA) workshop: On and Near-sensor Vision Processing, from Photons to Applications*, IEEE, 2021.

[112] Y. Liu, J. Chen, L. Bose, P. Dudek, and W. Mayol-Cuevas, “Bringing a robot simulator to the scamp vision system,” in *2021 IEEE International Conference on Robotics and Automation (ICRA) workshop: On and Near-sensor Vision Processing, from Photons to Applications*, IEEE, 2021.

[113] B. Guillard, *Optimising convolutional neural networks for super fast inference on focal-plane sensor-processor arrays*.
PhD thesis, Imperial College London, 2019.

[114] M. Wong, S. Saeedi, and P. H. Kelly, *Analog vision-neural network inference acceleration using analog SIMD computation in the focal plane*.
PhD thesis, Master’s thesis, Imperial College London-Department of Computing, 2018.

[115] J. Chen, Y. Liu, S. J Carey, and P. Dudek, “Proximity estimation using vision features computed on sensor,” in *International Conference on Robotics and Automation (ICRA)*, pp. 2689 – 2695, 31 May - 31 August 2020.

[116] U. Ozgunalp, R. Fan, X. Ai, and N. Dahnoun, “Multiple lane detection algorithm based on novel dense vanishing point estimation,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 3, pp. 621–632, 2017.

[117] H. Ni, Y. Liu, C. Zhang, Y. WANG, F. XIA, and Z. Qiu, “Sorting system algorithms based on machine vision for delta robot,” *Robot*, vol. 38, no. 1, pp. 49–55, 2016.

[118] M. A. Sotelo, F. J. Rodriguez, and L. Magdalena, “Virtuous: vision-based road transportation for unmanned operation on urban-like scenarios,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 5, pp. 69–83, June 2004.

[119] Y. Liu, R. Fan, B. Yu, M. J. Bocus, M. Liu, H. Ni, J. Fan, and S. Mao, “Mobile robot localisation and navigation using lego nxt and ultrasonic sensor,” in *2018 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pp. 1088–1093, IEEE, 2018.

[120] M. Mujahed, D. Fischer, and B. Mertsching, “Safe gap based (sg) reactive navigation for mobile robots,” in *Mobile Robots (ECMR), 2013 European Conference on*, pp. 325–330, IEEE, 2013.

[121] J. A. Tobaruela and A. O. Rodríguez, “Reactive navigation in extremely dense and highly intricate environments,” *PloS one*, vol. 12, no. 12, p. e0189008, 2017.

[122] B. Penin, P. R. Giordano, and F. Chaumette, “Vision-based reactive planning for aggressive target tracking while avoiding collisions and occlusions,” *IEEE Robotics and Automation Letters*, vol. 3, no. 4, pp. 3725–3732, 2018.

[123] F. Galluppi, C. Denk, M. C. Meiner, T. C. Stewart, L. A. Plana, C. Eliasmith, S. Furber, and J. Conradt, “Event-based neural computing on an autonomous mobile platform,” in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2862–2867, IEEE, 2014.

[124] B. T. Lopez and J. P. How, “Aggressive 3-d collision avoidance for high-speed navigation,” in *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pp. 5759–5765, IEEE, 2017.

[125] T. Hwu, J. Isbell, N. Oros, and J. Krichmar, “A self-driving robot using deep convolutional neural networks on neuromorphic hardware,” in *Neural Networks (IJCNN), 2017 International Joint Conference on*, pp. 635–641, IEEE, 2017.

[126] Y. Pan, C.-A. Cheng, K. Saigol, K. Lee, X. Yan, E. Theodorou, and B. Boots, “Agile autonomous driving using end-to-end deep imitation learning,” *Proceedings of Robotics: Science and Systems. Pittsburgh, Pennsylvania*, 2018.

[127] Y. Kong and Y. Fu, “Discriminative relational representation learning for rgb-d action recognition,” *IEEE Transactions on Image Processing*, vol. 25, pp. 2856–2865, June 2016.

[128] R. Fan and N. Dahnoun, “Real-time stereo vision-based lane detection system,” *Measurement Science and Technology*, vol. 29, no. 7, p. 074005, 2018.

[129] R. Fan, S. Duanmu, Y. Liu, Y. Zhu, J. Jiao, M. J. Bocus, Y. Yu, L. Wang, and M. Liu, "Real-time binocular vision implementation on an soc tms320c6678 dsp," in *International Conference on Computer Vision Systems*, pp. 13–23, Springer, 2019.

[130] R. Fan, H. Wang, P. Cai, and M. Liu, "Sne-roadseg: Incorporating surface normal information into semantic segmentation for accurate freespace detection," in *ECCV*, Springer, 2020.

[131] C. Greatwood, L. Bose, T. Richardson, W. Mayol-Cuevas, J. Chen, S. J. Carey, and P. Dudek, "Tracking control of a UAV with a parallel visual processor," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2017-Septe, pp. 4248–4254, 2017.

[132] J. Antich Tobaruela and A. Ortiz Rodriguez, "Reactive navigation in extremely dense and highly intricate environments," *PloS one*, vol. 12, no. 12, p. e0189008, 2017.

[133] D. Nakhaeinia, S. H. Tang, S. B. M. Noor, and O. Motlagh, "A review of control architectures for autonomous navigation of mobile robots," *International Journal of the Physical Sciences*, vol. 6, no. 2, pp. 169–174, 2011.

[134] J. G. Ziegler, N. B. Nichols, *et al.*, "Optimum settings for automatic controllers," *trans. ASME*, vol. 64, no. 11, 1942.

[135] T. Lindeberg, "A computational theory of visual receptive fields," *Biological Cybernetics*, vol. 107, pp. 589–635, Dec 2013.

[136] T. Lindeberg, "Image matching using generalized scale-space interest points," *Journal of Mathematical Imaging and Vision*, vol. 52, pp. 3–36, May 2015.

[137] J. Chen, S. J. Carey, and P. Dudek, "Feature Extraction using a Portable Vision System," 2017.

[138] F. Juefei-Xu and M. Savvides, "Subspace-based discrete transform encoded local binary patterns representations for robust periocular matching on nist's face recognition grand challenge," *IEEE transactions on image processing*, vol. 23, no. 8, pp. 3490–3505, 2014.

[139] S. Ke-Chen, Y. Yun-Hui, C. Wen-Hui, and X. Zhang, "Research and perspective on local binary pattern," *Acta Automatica Sinica*, vol. 39, no. 6, pp. 730–744, 2013.

[140] D. Olid, J. M. Fácil, and J. Civera, "Single-view place recognition under seasonal changes," *arXiv preprint arXiv:1808.06516*, 2018.

[141] M. Cummins and P. Newman, "Fab-map: Probabilistic localization and mapping in the space of appearance," *The International Journal of Robotics Research*, vol. 27, no. 6, pp. 647–665, 2008.

[142] R. Arroyo, P. F. Alcantarilla, L. M. Bergasa, and E. Romera, "Towards life-long visual localization using an efficient matching of binary sequences from images," in *2015 IEEE international conference on robotics and automation (ICRA)*, pp. 6328–6335, IEEE, 2015.

[143] Y. Liu and H. Zhang, "Visual loop closure detection with a compact image descriptor," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1051–1056, IEEE, 2012.

[144] N. Sünderhauf, P. Neubert, and P. Protzel, "Are we there yet? challenging seqslam on a 3000 km journey across all four seasons," in *Proc. of Workshop on Long-Term Autonomy, IEEE International Conference on Robotics and Automation (ICRA)*, p. 2013, 2013.

[145] P. Drews, G. Williams, B. Goldfain, E. A. Theodorou, and J. M. Rehg, "Aggressive deep driving: Combining convolutional neural networks and model predictive control," in *Proceedings of the 1st Annual Conference on Robot Learning*, vol. 78 of *Proceedings of Machine Learning Research*, pp. 133–142, 13–15 Nov 2017.

[146] M. Browne, S. S. Ghidary, and N. M. Mayer, "Convolutional neural networks for image processing with applications in mobile robotics," in *Speech, Audio, Image and Biomedical Signal Processing using Neural Networks*, pp. 327–349, Springer, 2008.

[147] Q. Liu, Z. Zhou, S. R. Shakya, P. Uduthalapally, M. Qiao, and A. H. Sung, "Smartphone sensor-based activity recognition by using machine learning and deep learning algorithms," *International Journal of Machine Learning and Computing*, vol. 8, no. 2, pp. 121–126, 2018.

[148] R. Zhao, X. Niu, Y. Wu, W. Luk, and Q. Liu, "Optimizing cnn-based object detection algorithms on embedded fpga platforms," in *Applied Reconfigurable Computing* (S. Wong, A. C. Beck, K. Bertels, and L. Carro, eds.), (Cham), pp. 255–267, Springer International Publishing, 2017.

[149] A. Aimar, H. Mostafa, E. Calabrese, A. Rios-Navarro, R. Tapiador-Morales, I. Lungu, M. Milde, F. Corradi, A. Linares-Barranco, S. Liu, *et al.*, "Nullhop: A flexible convolutional neural network accelerator based on sparse representations of feature maps.," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 3, pp. 644–656, 2019.

[150] R. Andri, L. Cavigelli, D. Rossi, and L. Benini, "Yodann: An architecture for ultralow power binary-weight cnn acceleration," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 48–60, 2017.

[151] Y.-H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energy-efficient reconfigurable accelerator for deep convolutional neural networks," *IEEE journal of solid-state circuits*, vol. 52, no. 1, pp. 127–138, 2016.

[152] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[153] Z. Lin, M. Courbariaux, R. Memisevic, and Y. Bengio, "Neural networks with few multiplications," *arXiv preprint arXiv:1510.03009*, 2015.

[154] Y. Kuang, "Deep neural network for deep sea plankton classification," tech. rep., Technical Report, 2015.

[155] J. Sim, J.-S. Park, M. Kim, D. Bae, Y. Choi, and L.-S. Kim, "A 1.42 tops/w deep convolutional neural network recognition processor for intelligent ioe systems," in *2016 IEEE International Solid-State Circuits Conference (ISSCC)*, pp. 264–265, IEEE, 2016.

[156] C. Zhu, K. Huang, S. Yang, Z. Zhu, H. Zhang, and H. Shen, "An efficient hardware accelerator for structured sparse convolutional neural networks on fpgas," *arXiv preprint arXiv:2001.01955*, 2020.

[157] K. Guo, L. Sui, J. Qiu, J. Yu, J. Wang, S. Yao, S. Han, Y. Wang, and H. Yang, "Angel-eye: A complete design flow for mapping cnn onto embedded fpga," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 35–47, 2017.

[158] G. Efland, S. Parikh, H. Sanghavi, and A. Farooqui, "High performance dsp for vision, imaging and neural networks.," in *Hot Chips Symposium*, pp. 1–30, 2016.

[159] B. Sun, C. Milpitas, D. Liu, L. Yu, J. Li, H. Liu, W. Zhang, and T. Torng, “System demonstration of mram co-designed processing-in-memory cnn accelerator for mobile and iot applications,”

[160] N. P. Jouppi, C. Young, N. Patil, D. Patterson, G. Agrawal, R. Bajwa, S. Bates, S. Bhatia, N. Boden, A. Borchers, *et al.*, “In-datacenter performance analysis of a tensor processing unit,” in *Proceedings of the 44th Annual International Symposium on Computer Architecture*, pp. 1–12, 2017.

[161] Y. Zhu, M. Mattina, and P. Whatmough, “Mobile machine learning hardware at arm: a systems-on-chip (soc) perspective,” *arXiv preprint arXiv:1801.06274*, 2018.

[162] A. Jimenez-Marrufo, A. Mendizabal, S. Morillas-Castillo, R. Dominguez-Castro, S. Espejo, R. Romay-Juarez, and A. Rodriguez-Vazquez, “Data matrix code recognition using the eye-ris vision system,” in *2007 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1214–1214, IEEE, 2007.

[163] F. Karabiber, P. Arena, L. Fortuna, S. De Fiore, G. Vagliasindi, and S. Arik, “Implementation of a moving target tracking algorithm using eye-ris vision system on a mobile robot,” *Journal of Signal Processing Systems*, vol. 64, no. 3, pp. 447–455, 2011.

[164] L. Nicolosi, R. Tetzlaff, F. Abt, A. Blug, and H. Höfler, “Cellular neural network (cnn) based control algorithms for omnidirectional laser welding processes: Experimental results,” in *2010 12th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA 2010)*, pp. 1–6, IEEE, 2010.

[165] L. C. Gontard, R. Carmona-Galán, and Á. Rodríguez-Vázquez, “Cellular-neural-network focal-plane processor as pre-processor for convnet inference,” in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, IEEE, 2020.

[166] J. Yang, X. Shen, J. Xing, X. Tian, H. Li, B. Deng, J. Huang, and X.-s. Hua, “Quantization networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7308–7316, 2019.

[167] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, “What is the state of neural network pruning?,” *arXiv preprint arXiv:2003.03033*, 2020.

[168] Y. Guo, “A survey on methods and theories of quantized neural networks,” *arXiv preprint arXiv:1808.04752*, 2018.

[169] M. Mathew, K. Desappan, P. Kumar Swami, and S. Nagori, “Sparse, quantized, full frame cnn for low power embedded devices,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 11–19, 2017.

[170] E. C. Orenstein, O. Beijbom, E. E. Peacock, and H. M. Sosik, “Whoi-plankton-a large scale fine grained visual recognition benchmark dataset for plankton classification,” *arXiv preprint arXiv:1510.00745*, 2015.

[171] P. Dudek, “Accuracy and efficiency of grey-level image filtering on vlsi cellular processor arrays,” in *Proc. CNNA*, pp. 123–128, 2004.

[172] S. S. Rautaray and A. Agrawal, “Vision based hand gesture recognition for human computer interaction: a survey,” *Artificial intelligence review*, vol. 43, no. 1, pp. 1–54, 2015.

[173] Y. LeCun, C. Cortes, and C. Burges, “Mnist handwritten digit database,” *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, vol. 2, p. 18, 2010.

[174] T. Simons and D.-J. Lee, “A review of binarized neural networks,” *Electronics*, vol. 8, no. 6, p. 661, 2019.

[175] L. Bose, P. Dudek, J. Chen, S. J. Carey, and W. W. Mayol-Cuevas, “Fully embedding fast convolutional networks on pixel processor arrays,” in *European Conference on Computer Vision – ECCV 2020*.

[176] E. Sari, M. Belbahri, and V. P. Nia, “How does batch normalization help binary training,” *arXiv preprint arXiv:1909.09139*, 2019.

[177] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” *arXiv preprint arXiv:1502.03167*, 2015.

[178] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, “Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1,” *arXiv preprint arXiv:1602.02830*, 2016.

[179] Z. Liu, Z. Shen, M. Savvides, and K.-T. Cheng, “Reactnet: Towards precise binary neural network with generalized activation functions,” in *European Conference on Computer Vision*, pp. 143–159, Springer, 2020.

[180] P. Yin, J. Lyu, S. Zhang, S. Osher, Y. Qi, and J. Xin, “Understanding straight-through estimator in training activation quantized neural nets,” *arXiv preprint arXiv:1903.05662*, 2019.

[181] X. Wang, M. Kan, S. Shan, and X. Chen, “Fully learnable group convolution for acceleration of deep neural networks,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[182] L. Bose, P. Dudek, J. Chen, and S. J. Carey, “Sand castle summation for pixel processor arrays,” in *The 17th International Workshop on Cellular Nanoscale Networks and their Applications, CNNA*, 2021.

[183] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik, “Emnist: Extending mnist to handwritten letters,” in *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 2921–2926, IEEE, 2017.

[184] C. Ding and X. He, “K-means clustering via principal component analysis,” in *Proceedings of the twenty-first international conference on Machine learning*, p. 29, 2004.

[185] J. Dai, Y. Li, K. He, and J. Sun, “R-fcn: Object detection via region-based fully convolutional networks,” *arXiv preprint arXiv:1605.06409*, 2016.

[186] G. Fu, C. Liu, R. Zhou, T. Sun, and Q. Zhang, “Classification for high resolution remote sensing imagery using a fully convolutional network,” *Remote Sensing*, vol. 9, no. 5, p. 498, 2017.

[187] J. Long, E. Shelhamer, and T. Darrell, “Fully convolutional networks for semantic segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[188] V. Iglovikov, S. Seferbekov, A. Buslaev, and A. Shvets, “Ternausnetv2: Fully convolutional network for instance segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 233–237, 2018.

[189] P. Dudek and P. J. Hicks, “A general-purpose processor-per-pixel analog simd vision chip,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 52, no. 1, pp. 13–20, 2005.

[190] F. Chollet, “Xception: Deep learning with depthwise separable convolutions,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1251–1258, 2017.

[191] O. Michel, “Cyberbotics ltd. webots™: professional mobile robot simulation,” *International Journal of Advanced Robotic Systems*, vol. 1, no. 1, p. 5, 2004.

[192] W. Fuhl, G. Kasneci, and E. Kasneci, “Teyed: Over 20 million real-world eye images with pupil, eyelid, and iris 2d and 3d segmentations, 2d and 3d landmarks, 3d eyeball, gaze vector, and eye movement types,” *arXiv preprint arXiv:2102.02115*, 2021.

[193] O. Sener and V. Koltun, “Multi-task learning as multi-objective optimization,” *arXiv preprint arXiv:1810.04650*, 2018.

[194] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[195] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, “The graph neural network model,” *IEEE transactions on neural networks*, vol. 20, no. 1, pp. 61–80, 2008.

[196] G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, “Recent advances in physical reservoir computing: A review,” *Neural Networks*, vol. 115, pp. 100–123, 2019.

[197] O. Yilmaz, “Reservoir computing using cellular automata,” *arXiv preprint arXiv:1410.0162*, 2014.

[198] J. Chen, Y. Liu, S. J. Carey, and P. Dudek, “Proximity estimation using vision features computed on sensor,” in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2689–2695, 2020.

[199] H. Castillo-Elizalde, Y. Liu, L. Bose, and W. Mayol-Cuevas, “Weighted node mapping and localisation on a pixel processor array,” in *IEEE International Conference on Robotics and Automation (ICRA), Xi’an, China*, May, 2021.

[200] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1321–1326, IEEE, 2013.

[201] S. James, M. Freese, and A. J. Davison, "Pyrep: Bringing v-rep to deep robot learning," *arXiv preprint arXiv:1906.11176*, 2019.

[202] W. Fan, Y. Liu, and Y. Xing, "Fully-simulated integration of scamp5d vision system and robot simulator," *arXiv preprint arXiv:2110.06386*, 2021.

[203] E. J. Stow, "Automatic code generation for simultaneous convolutional kernels on cellular processor arrays," 2020.

[204] E. Stow and P. H. J. Kelly, "Super-pixel compilation for pixel processor arrays," in *2021 IEEE International Conference on Robotics and Automation (ICRA) workshop: On and Near-sensor Vision Processing, from Photons to Applications*, IEEE, 2021.