ALMA MATER STUDIORUM - UNIVERSITÀ DI BOLOGNA

Dottorato di Ricerca in

Ingegneria Elettronica, delle Comunicazioni e Tecnologie dell'Informazione

---

Ciclo XXIX

ELECTRONIC SYSTEMS WITH HIGH ENERGY EFFICIENCY FOR EMBEDDED COMPUTER VISION

Relatore:

Chiar.mo Prof. Luca Benini

Candidato:

Dott. Francesco Paci

Correlatori:

Chiar.ma Prof.ssa Rita Cucchiara

Chiar.ma Prof.ssa Michela Milano

Esame finale anno 2017

To my father Luciano

and my mother Anna Maria

Contents

List of Figures

List of Tables

Abstract

Electronic systems are now widely adopted in everyday use. Moreover, nowadays there is an extensive use of embedded wearable and portable devices from industrial to consumer applications. The growing demand of embedded devices and applications has opened several new research fields due to the need of low power consumption and real time responsiveness. Focusing on this class of devices, computer vision algorithms are a challenging application target. In embedded computer vision hardware and software design have to interact to meet application specific requirements. The focus of this thesis is to study computer vision algorithms for embedded systems. The presented work starts presenting a novel algorithm for an IoT stationary use case targeting a high-end embedded device class, where power can be supplied to the platform through wires. Moreover, further contributions focus on algorithmic design and optimization on low and ultra-low power devices. Solutions are presented to gesture recognition and context change detection for wearable devices, focusing on first person wearable devices (Ego-Centric Vision), with the aim to exploit more constrained systems in terms of available power budget and computational resources. A novel gesture recognition algorithm is presented that improves state of art approaches. We then demonstrate the effectiveness of low resolution images exploitation in context change detection with real world ultra-low power imagers. The last part of the thesis deals with more flexible software models to support multiple applications linked at runtime and executed on Cortex-M device class, supporting critical isolation features typical of virtualization-ready

CPUs on low-cost low-power microcontrollers and covering some defects in security and deployment capabilities of current firmwares.

Introduction

Background

Electronic Systems are nowadays widespread and aid the human beings in most of the every-day activities. Moreover the growth in energy efficiency over performance of CPUs has driven the design of systems capable to be embedded in small and portable form factors and to be integrated with peripherals.

Embedded systems have evolved to meet several requirements among the years: *portability, energy efficiency, peak performance, costs*. Nowadays Embedded Systems integrate from low and ultra-low power low-end Microcontroller Units (MCUs) that consume power in the order of the mWatt to high-end SoCs that target higher performance and have higher power needs. Moreover they expose different processing capabilities, they can embed a single core or expose multiples or many cores. This broad range of different devices enables to target many application domains, such as automotive and industrial systems (MCU and FPGA), communication systems (DSP and FPGA), and computer vision.

Computer Vision (CV) is the ability of extract information from images and videos and the information extracted can be used to take decisions, reproducing the human vision system. The research in this field

has been started since early 1970s [1], with works in robotic and artificial intelligence.

The marriage of computer vision and embedded systems has some early examples coming from 1980s. As stated in [2] one of the first examples of Embedded Computer Vision (ECV) system is the Xerox optical mouse. In this primitive, but successful, example the acquisition system is composed by a really simple optical sensor equipped with an array of only $4x4$ pixels and the embedded systems is an Application Specific Integrated Circuit (ASIC). This is one of the first cases where computer vision engineers and hardware designers had to cooperate to create an embedded computer vision system. Since this successful case, thanks to the growth in potability and autonomy of embedded systems, as well as the system integration, computer vision on embedded systems had a massive diffusion in research and industry. Over the years computer vision and embedded systems have evolved together to meet and create new applicative contexts.

Embedded Computer Vision

Nowadays applications of Embedded Computer Vision are diverse. Implementation ranges from human assistance, like autonomous devices or robots [3], to video games [4] or virtual reality systems [5]. Automotive is one field in which ECV is very active: pedestrian detection [6], lane departure prevention [7, 8] or obstacle detection [9]. Smart surveillance uses computer vision to pre-filter videos, examples include vehicle traffic monitoring [10], event surveillance and analysis of human features [11].

Because of the diversity of these applications several constraints must

be taken into account bringing computer vision to embedded systems. In particular *Power Budget*, *Physical Size*, *Responsiveness*, *Flexibility*, *Accuracy* and *Costs* are most influencing the design of ECV systems.

Power Budget is a first important requirement that must be taken into account. There are scenarios like industrial assembly lines where this is a more relaxed constraint, while it is very important when devices are battery operated. *Physical Size* is influencing the design when embedded devices are portable, this means not only the target size in terms of mm^3 but also the form factor. This constraint is important in novel scenarios like wearable devices, where the device has to be worn so it must also be comfortable for the final user. The third important constraint is the *Responsiveness* of the system. There are computer vision systems whose results are not expected to be produced in few milliseconds but images or videos can be elaborated in time. While, for instance, in collision avoidance systems the video stream must be evaluated in hard time deadline to enable actuators. *Flexibility* can be meant in terms of code deployment, ease of update, as well as in terms of programmability of the hardware platform. The last two constraints are *Accuracy* and *Costs*. *Accuracy* must be taken into account more carefully in solutions like Biometric Analysis Systems, where the security or the health of the patient is involved, while *Costs* are important in distributed scenarios where deployment of several devices is required.

All these constraints are not to be considered stand-alone but must cooperate to meet the requirements of the specific use-case.

Several design choices can be employed to meet outlined constraints. We summarize them into 4 categories: choice of the *hardware platform*, *low-level software optimizations*, *high-level software optimizations*, *design of new algorithms*.

A first important choice is the *hardware platform*. Field Programmable Gate Arrays (FPGAs) have been deeply studied as a solution for Embedded Vision Systems [12]. The advantages of using these platforms are power efficiency and the possibility to use automatic tools to export an FPGA design to an ASIC. While, the main difficulty is the programmabilty that usually requires longer time and advanced skills. Most applications in computer vision implemented on FPGAs deal with stereo camera matching and several old and new works have been proposed [13, 14]. Other solutions rely on more flexible devices based on ARM Cortex SoC, these platforms range from single to multiple cores and, for power efficiency, they enable CPU frequency scaling and embed heterogeneous cores. They are preferred to FPGAs because they permit fast prototyping and higher software flexibility. They can be coupled with multi and many-core accelerators like GPUs or VLIW Digital Signal Processors (DSPs). Examples are Movidius Myriad [15], TI AccelerationPAC [16].

A clear trend in last years is the exploitation of low-end platforms [17]. There are, as well, new commercial platforms for fast prototyping [18]. Such devices find their application mostly in robotics, i.e. drones or unmanned vehicles and the Internet of Things (IoT). One novel application for low-power and ultra-low-power platforms is visual life-logging [19].

The choice of the hardware platform not only means the computational unit but also the choice of the Vision Sensor (imager). In last

years works explore the usage of low-power and ultra-low-power imagers [20–22], not only measuring pixel intensity but binary patterns as well.

In the software domain a first solution that can be employed is *Low-Level Software Optimizations*, re-writing routines to exploit the specific hardware [23]. Optimizations range from exploiting core parallelism to vector units or the coupled hardware accelerators when available.

High-Level Software Optimizations are another viable solution. The target of this technique is to minimize the number of serial operation that the algorithm tailored for a specif task requires. This can be done by re-design the algorithm to find a simpler and less resource-hungry solution. We have several notable and well known recent [24] and less recent [25–29] works that use this technique to reduce the computational load.

Design of New Algorithms targeted for a novel or an existing scenario is the third software way to bring computer vision to embedded systems. An example of novel scenario is wearable systems for First Person Vision also called ego-vision. There are several very recent works that range from Action [30] and Activity Recognition [31,32] to Attention Prediction [33], Engagement Detection [34], Visual Saliency [35]. On the other hand new solutions are provided to existing problems with the diffusion of machine learned features. Deep learning [36] is driving the research in computer vision in the last years and improving existing solutions in terms of accuracy and performance.

Thesis Contributions

Open Challenges include the study of CV algorithms on integrated embedded platforms, providing the accuracy of the proposed solution along with an analysis of the performance in a possible target scenario. Moreover the study of reliable and affordable solutions finding the best tradeoff between requirements and constraints is, in some scenarios, a missing critical link between computer vision and embedded systems. The study of deep learning algorithms on low resolution imagers has only been partially explored. Computer Vision is widely adopting deep learning. The study of neural networks on this class of imagers would open new perspectives in space and power constrained low-end scenarios. On low-end platforms, moreover, there is a need of flexible and secure software solutions for smart application deployment. In this thesis we deal with all these challenges presenting the contributions in three scenarios of ECV on electronic systems with high energy efficiency with growing requirements and constraints.

The first major contribution of this thesis is the design of a novel algorithm for an image based IoT Node. To meet the requirements we started with the assumption that the stationarity of the node can be exploited to present a solution with a better tradeoff between accuracy and operation per second required. In terms of computer vision optimization, we present the design of a novel algorithm for a task that already has some solutions in literature but these solutions have higher power and performance requirements, that are not suitable for an embedded system.

From a high-end embedded system based on android in a stationary condition, we move to a second scenario where the embedded system is

of the same power/performance class but in wearable context.

The second major contribution of this thesis is a novel gesture recognition algorithm for ego-vision applications that uses trajectories, appearance features and hand segmentation to classify static and dynamic hand movements and that can achieve high accuracy results even when trained with a few positive samples. Then we present a performance analysis of the algorithm on a workstation class x86 machine.

The static and dynamic hand gesture recognition algorithm has been fully integrated in wearable vision system, this is the third major contribution. A distributed architecture that improves museum visitors' experience composed by ego-vision wearable devices and a central server, and it is capable of recognizing users' gestures and artworks. Thus we present a performance evaluation of our algorithm on an ARM big.LITTLE heterogeneous platform and an implemented tradeoff between accuracy and performance.

For the third scenario presented in this thesis we set in the same wearable context discussed in the previous one but studying the exploitation of a low-end (MCU) class of devices.

We then explore how, even with very limited resolution, we can obtain context awareness and understand, at least, a change of context in our day-life. Dealing with a novel image acquisition system (imager) we study the exploitation of this imager designing a new algorithm based on convolutional neural networks. **The forth major contribution is a context change detector for low-resolution images based on a wearable ego-centric camera with ultra-low power consumption**.

The fifth major contribution of the thesis is a more flexible software model to support multiple applications runtime

linked and executed on Cortex-M class of devices, supporting critical isolation features typical of virtualization-ready CPUs on low-cost low-power microcontrollers and covering some defects in security and deployment capabilities of current firmwares.

Thesis Overview

The thesis is divided into 5 chapters. In Figure 1 we show their positioning based on computer vision and embedded systems features.

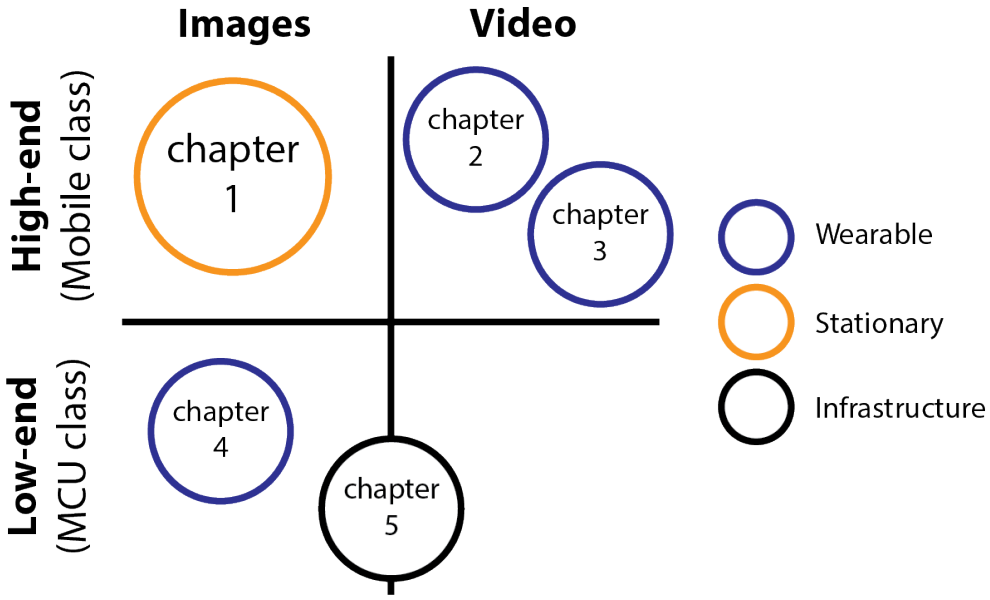


Figure 1: Structure of the thesis.

Chapter 1 presents an occupancy monitoring algorithm based on images in a stationary setting. Chapter 2 shows a novel monocular gesture recognition algorithm for first person vision, and Chapter 3 shows how this algorithm can be integrated in an embedded system using wearable vision sensors to enhance visitors' museum experiences. Chapter 4 deals with an ultra-low power low-resolution imager presenting a novel first

person context change detector, while Chapter 5 deals with some infrastructure defects, presenting a more flexible and secure software model to support multiple applications runtime linked and executed on low-end device class.

Embedded Computer Vision Metrics

A clarification on the evaluation metrics of an ECV system is given in this subsection. To evaluate a computer vision system four different metrics must be considered, as reported in [1]:

M.1 Application Performance

Percentage of success in doing the task

M.2 Speed

Time to complete the task

M.3 Power Dissipation

The energy required by the embedded system to execute the task

M.4 System Size

The physical size of the embedded system

In this thesis we choose a nomenclature closer to the embedded system world than to the CV world. We will refer to **M.1** with the term *accuracy*, this is more often used in the Embedded Systems to refer to the quality of the elaboration of the algorithm. The term *performance* will instead be used for **M.2**, denoting the speed of elaboration of the solution provided. The *power dissipation* **M.3** and the *size* **M.4** of the solution cannot be interpreted with any ambiguity.

Chapter 1

Image Based IoT Node for Classroom Occupancy Monitoring

1.1 Overview

In this first chapter we present an IoT node which uses the processing of images taken with camera nodes to provide occupancy information in mostly-static contexts. The task poses a first challenge because of the limited computational resources of the embedded platform, while the stationary setting of the vision nodes provides an easier and more relaxed design constraint that permits to scale down sampling rate and process images instead of videos. After designing the image based algorithm the IoT CV node has been further integrated in a heterogeneous WSN, consisting of sensor nodes and camera nodes for building monitoring. Indeed, computer vision typically requires notable amounts of complexity, performance and power consumption levels not compatible with a distributed

wireless sensing scenario.

By combining local processing, low power hardware design and power management we will prove the efficiency of our approach and we will demonstrate, by experimental results the feasibility of occupancy monitoring in a real deployment scenario, with acceptable levels of power consumption and accuracy.

The chapter is structured as follows. First related works are presented. Section 1.3 deals with the design of the people counting algorithm. Then we present the image based node. In Section 1.4 the heterogeneous sensors network is presented. In Section 1.5 results in terms of accuracy and power consumption are shown and discussed. Finally, Section 1.6 concludes the chapter.

1.2 Related Work

Internet of Things and Wireless Sensor Networks are becoming a mature technology and several structural and building monitoring real systems based on wireless sensor networks have been designed [37] and are already on the market, for instance IBM TRIRIGA® Energy Optimization [38] and Siemens Desigo [39]. Such systems allow a flexible room automation with a fine-grained control and monitoring thanks to wireless nodes capable to measure environmental conditions and energy consumption [40]. The centralized control rooms are equipped with timers and there are several thermostats placed all over the building, but those systems miss a complete automated system for heating control based on room occupancy.

The solution that we provide focuses on occupancy monitoring by a

distributed on-node algorithm. Several works have already investigated the trade-off between central vs. distributed processing in a WSN, such as [41–43]. Focusing on the people counting task, there are several applications that try to implement an efficient solution using a variety of sensors or distributed algorithms on computing platforms. A big part of the literature relies on video cameras stream processing. There are good results achieved by [44] [45]. They track people in video, using different methods, as a consequence these systems could be used to enumerate and then count the number of people in a scene. However they need a high computational effort to achieve it.

People detection in crowded scenes tries to estimate the number of people by building occupancy models. Several works has been presented on this topic. Li *et al.* [46] give an in depth description covering different aspects of scene analysis, including crowd motion pattern learning, crowd behavior and activity analyses, and anomaly detection in crowds. They do not explicitly cover People Counting though. Another less recent survey on people counting by Zhan *et al.* [47] gives a description of crowd density measurement which can be done by counting people in the scene. An interesting work is presented by Hou *et al.* [48], but it mostly deals with very crowded outdoor scenarios. More recent works, in this field, exploit new neural network techniques. For example good results are achieved in [49] where accuracy and performance are deeply studied but performance are not compatible with an embedded implementation. Moreover there are two works that should be noted [50, 51]. They have as main target counting people in indoor spaces, so they try to address the problem similarly to our approach. The main difference is a lack of performance assessment and accuracy information to make a comparison

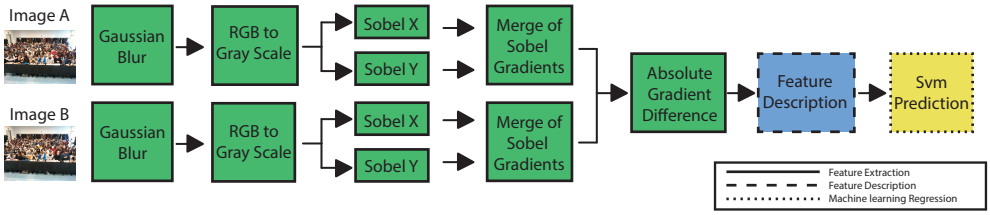


Figure 1.1: Steps of Entropy based Algorithm.

with our results.

In WSN, the applications for counting people with a camera mostly focus on ceiling top down view. The work proposed by Lee, Tsai and Hung [52] defines an interest region that is used like a gate, an intereste region to counts the poeple crossing it, and uses a Gaussian Mixture Model (GMM) [53] to isolate the foreground from the background. While the work by Teixera and Saviddes [54] tries to count the number of people in the field of view using a single wide-angle camera. They use size and movements of humans to recognize them, calculating a motion histogram. It has limited scalability due to their need for several camera sensors to cover a large area.

1.3 People Counting: An Entropy Based Approach

In this section we describe in detail the algorithm developed for counting people on the visual nodes. The target of this algorithm is to find the number of people inside classrooms during lectures. Our approach is based on one assumption: most of the changes that take place inside a room are related to human activity. We want moreover to be robust to these two kinds of changes:

1. **Slow Changes** related to environment or sunlight (i.e. weather/time-of-day related light changes). These changes take place in minutes or hours.

2. **Instantaneous Changes** related to the lighting system (i.e. when light is switched on or off).

Starting from this idea, with a wall-mounted still camera, we decided to scale down the sampling rate (intervals between subsequent frames) and take picture in pairs at a fixed time interval. Exploiting the changes observed in the two pictures of a pair, we can correlate them with the number of people inside the classroom, measuring the entropy produced within the interval. After extensive experimentation we settled that a good interval to track entropy produced by people (even sitting in the same place for a long time) is to take pictures in pairs outdistanced of 60 seconds and look at the differences that occurs within them. This reduces the sensitivity of *Slow Changes* because they take place over longer time scales. Elaborating from 15 to 25 images per second, as mostly of the on-the-market camera sensors can provide, is moreover a big overkill of data, not needed for our purpose. To further reduce the sensitivity with respect to *Slow Changes* we decided to work in gradient space, that is more robust to small light changes than normal color/light space.

Instantaneous changes cannot be tolerated with the approach described above. Hence, a large gradient variation is expected when an *Instantaneous Change* happens in the 60 seconds interval between two images. On the other hand, these changes are less frequent than *Slow Changes* and the interval of 60 seconds that we chose for images pairs helps reducing even more their impact. To remove the false counts created by these changes, we decided add a filter that checks the differences

in between the pairs and if it is higher than the entropy measured with the maximum number of people inside the classroom, we invalidate the results and we take another picture. Then we elaborate this new picture with the second one of the original pair.

In next subsections, we describe each of the three parts that compose our algorithm: the first one deals with the entropy extraction, the second one with features description, and the third one explains the learning phase that matches described features with the numerical output of the algorithm.

1.3.1 Entropy Based Features

The main idea of the entropy based features extraction is to correlate two maps of gradients of two subsequent images, to find the features that characterize the presence of people. The choice of using gradients is due to the idea that people activity alters the original map of gradients and this activity can be correlated to the number of people in the room. Moreover gradients permit to avoid small light changes between the pairs of images.

The input for the gradients extraction are two color images sampled with 60 seconds of delay. This way the correlation of gradients of two images does not contain the difference between the empty room and the people that is on the foreground. It contains just what has been changed within the pair of images. This design presents multiple advantages. It prevents that Slow Activity affects the results of the algorithm (as described before) and, not less important, even if the configuration of the room (arrangement of furniture, chairs, desks) changes the results are not affected. Moreover gradients are less affected by fluctuation of

brightness due to exposure or small light changes.

The Absolute gradient difference depicted in Fig. 1.1 is preceded by four steps. The first one is a Gaussian blur with a 3×3 kernel. This step is done to clean the noise introduced from the image sensor. Then a conversion to gray scale is done for the gradients extraction.

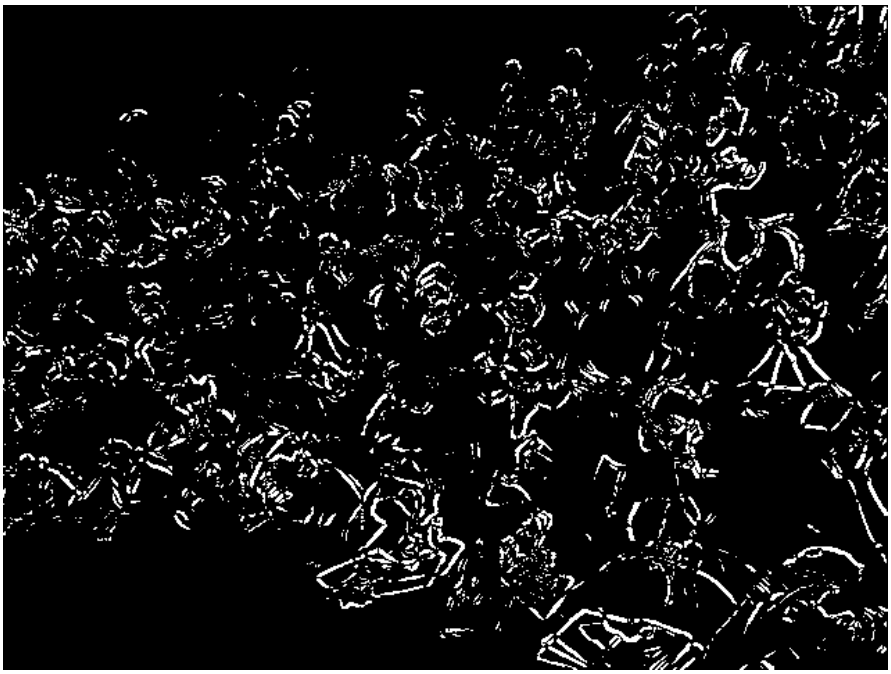
The extraction of gradient information from the images is done with two Sobel Kernels. In order to extract the x and y derivatives, these two convolutional kernels are used:

$$K_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}; \qquad K_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

This step is done for each of the two images. As a result of this step, two new matrices are produced, one for the x and one for the y derivative: K_x and K_y. To have a merged map for each image, K_x and K_y are transformed in the corresponding Modulus matrices and then an addition is performed, with a 0.5 weight for each matrix.

Finally to have a map of the features that describes the activity in the room, the results of the previous matrices are modulus subtracted (Absolute Gradient Difference). This operation removes all the gradient map's features that belong to the background, and only what has changed during the elapsed time between the two picture is described after this step. We will refer to this map in next section naming it Entropy Features Map (EFM), an example in Figure 1.2.

(a)



(b)

Figure 1.2: (a) One of the two images of one couple, (b) a binarized EMF

1.3.2 Features Description

The key idea of features description is simple: the area taken up by each person is related to the vertical position in the image space. People that sit in the first seats occupy a bigger area in the picture, while people in the back a smaller one.

Since the EFM contains information on people related to their size, we decided to make a description of the EFM by creating many accumulators, one for each horizontal line of EFM. So given D_x and D_y the horizontal and vertical size of EFM, there are D_y accumulators:

$$Acc_0 = \sum_{i=0}^{D_{x-1}} \begin{cases} 1 & \text{if } EMF(0,i) \text{ is foreground} \\ 0 & \text{if } EMF(0,i) \text{ is background} \end{cases}$$

...

...

...

$$Acc_{D_{y-1}} = \sum_{i=0}^{D_{x-1}} \begin{cases} 1 & \text{if } EMF(D_{y-1},i) \text{ is foreground} \\ 0 & \text{if } EMF(D_{y-1},i) \text{ is background} \end{cases}$$

Each one of this accumulators describe the number of foreground pixels for each line of the EFM. This way is possible for the learning phase to weight the foreground pixels of the image by line.

1.3.3 Characterization of a Room

Starting from the previous feature description, a characterization between the features description and the real number of people inside the classrooms is required. For this purpose, we decided to use a machine

learning stage that inputs the previous accumulators and learns the number of people from that data. We set up a Support Vector Machine and, since the output is an integer number belonging to the range 0 to 150 (number of people), we used a regression model. After different trials we choose ν-SVM and Radial Basis as most suitable SVM Model and Kernel, whose performance are discussed in the next Section.

1.4 Heterogeneous Sensors Network

In this section two kinds of nodes that compose the WSN are presented: the camera one and the sensors equipped one.

1.4.1 Image Based IoT Nodes

The camera nodes that we target are Android platforms (e.g. smartphones with embedded camera or development platforms with external camera). The network deployment has been done on Samsung Galaxy Pocket Smartphones. They are equipped with Android 4.0.4 and through a WIFI module they communicate directly to the main server (a cloud based server that collects data from all sensors and camera nodes). We use these devices to process images from building's rooms on board. For this purpose we designed and implemented an Android application that makes use of native C libraries for the computational back-end and of java libraries for the interface and control front-end. Thus, the result of processing is sent trough the network using TCP-IP and JSON based protocols to the centralized server. The application is designed to collect images at specific time intervals, that can be set by a graphical user interface. It is possible to send through the network either results from

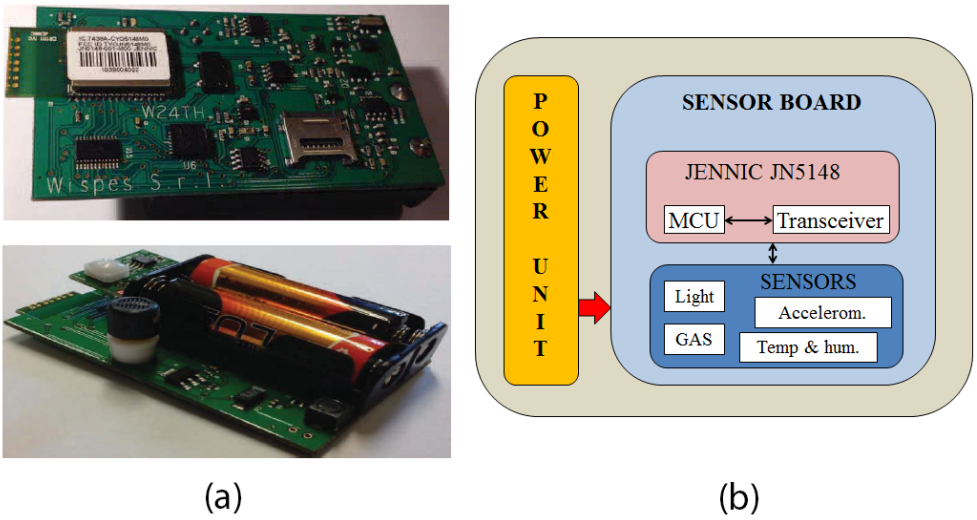


Figure 1.3: (a) W24TH Sensor node used for environmental monitoring and (b) block diagram of sensor design.

the local processing algorithm that is described in Section 1.3 or images to collect a dataset. The images are sent trough Secure Copy Protocol (SCP) and a fault tolerance module that permits long offline periods has been implemented. The collection of images can be stopped by a timer to, for instance, disable image sampling overnight or during weekends. Furthermore there is a module for alerts, designed to advice if there are power supply issues.

1.4.2 Sensor Nodes

The wireless sensor nodes used for environmental monitoring is a W24TH model, developed by *Wispes srl* [55]. The node (Fig. 1.3) is designed to minimize the energy wasted during idle state and the used CPU is a 32-bit architecture (NXP JN5148) module [56] with ultra-low-power features and high performance wireless capability targeted at 802.15.4 networking

applications. Compared to other similar and commercial sensor nodes [57], it features about 35% power savings with a power consumption of $35mW$ for TX mode and $42mW$ for RX, while it can switch to a sleep states with only $8\mu A$ consumption. The most remarkable characteristics are the 32-bit computation capability at $32MHz$ useful to perform on-line processing. They are equipped with several sensors: temperature, relative humidity, light and dock for the catalytic MOX sensors.

1.4.3 Data Collection

The camera nodes are directly connected to the internet trough a WIFI module and they can communicate elaborated results directly to cloud servers. They transmit packets over the network trough TCP-IP protocol encapsulating data using JSON format.

The sensor nodes are organized as a tree network where the root sensor collects data from the other nodes, then communicate collected values through a gateway.

The information sampled from sensors and cameras are gathered together in a cloud based server. This will make possible to show the retrieved values and to further trigger remote actions.

1.5 Experimental Results

In this section we present results in terms of accuracy, performance and energy. For this purpose we collected a dataset of images sized 1600×1200 px, taken from a frontal viewpoint (Fig. 1.4) from two different classrooms.

Figure 1.4: Four dataset images, two from Room 1 (on top) and two from Room 2 (on bottom).

1.5.1 Algorithm Detection Accuracy

The dataset for the learning phase consists, for each room, of four days of images, sampled in pairs. Each pair consists of two images sampled within 60 seconds interval. The pairs are sampled with 10 minutes interval. The sampling started at 08:00 a.m. and went on until 08:00 p.m., so the total number of images per day is 144, and the total images for the learning phase is 576. With these images we trained the SVM learning phase described is Section 1.3.

In figure 1.5 we show the results of a test set consisting of 144 images for each room. It depicts the real number of people and the detected one. Each detection is related to the elaboration of one pair. Ground

truth has been human calculated and thus it could contain errors since counting people in a room from pictures is a difficult task even for human beings, due to partial or total occlusions and background cluttering. Moreover the ground truth is relative to the second picture of the pair. Watching the depicted results, the detection line follows ground truth, except sometimes that is possible to notice a lead or lag of one measurement. This is due to higher entropy when people enter and leave rooms. For instance it can be noticed an error in the pair 160543 within Room 1 or the pair 190543 within Room 2 of figure 1.5.

The error calculation has been done with equation (1) and (2): MAE (Mean Absolute Error) and RMSE (Root Mean Square Error) respectively:

$$MAE = \frac{\sum_{i=0}^{N-1} \left| R_i - D_i \right|}{N} \tag{1.1}$$

$$RMSE = \sqrt{\frac{\sum_{i=0}^{N-1} \left(R_i - D_i \right)^2}{N}} \tag{1.2}$$

Where N is total number of test images, R and D the real and detected number of people respectively.

The error related to the dataset composed of 5 days (four for training and one for test) is:

Best Result:

Room 1: $MAE = 04.86$ $RMSE = 09.39$

Room 2: $MAE = 04.98$ $RMSE = 07.12$

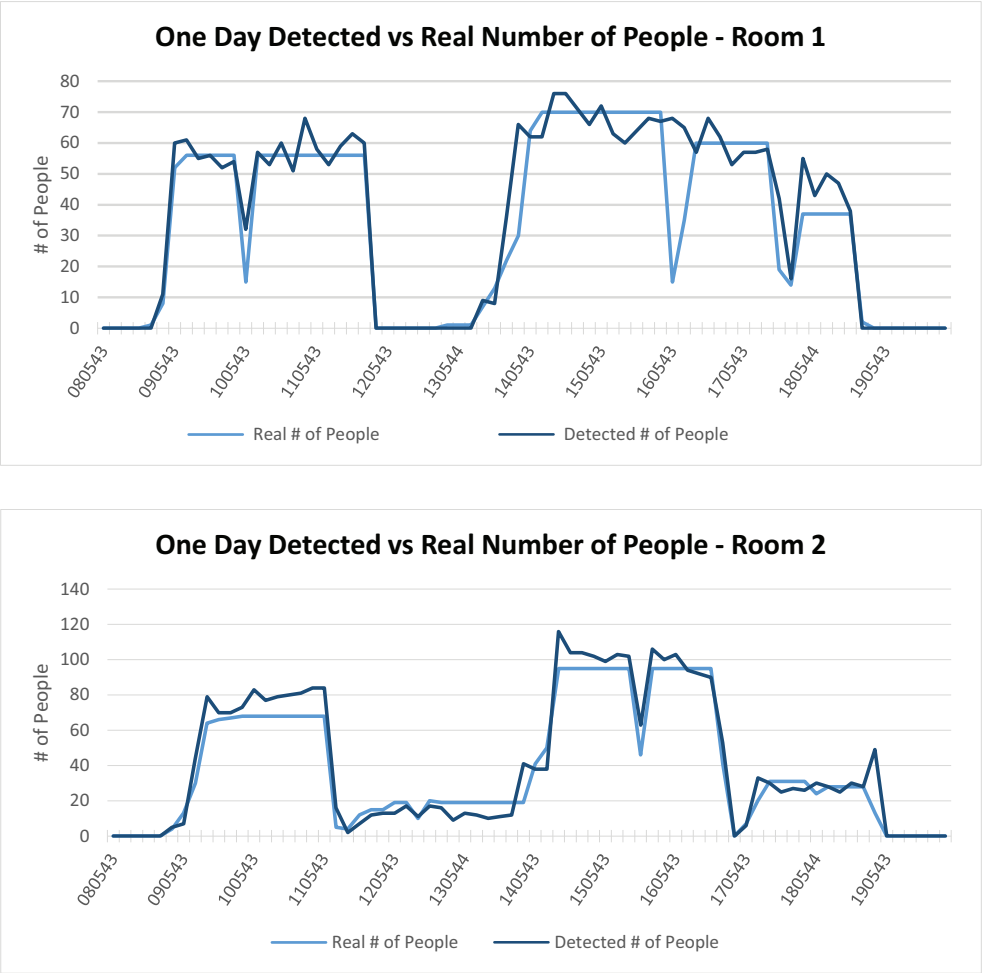


Figure 1.5: Comparison of detected people with ground truth during a whole day. X axis represents time of the taken picture (total number of pictures per day: 72 pairs), and Y axis the number of detected people.

Worst result:

$$\text{Room 1:} \qquad MAE = 09.44 \qquad RMSE = 15.71$$

$$\text{Room 2:} \qquad MAE = 08.38 \qquad RMSE = 12.24$$

The four days of training and the day of test have been randomly chosen, this is the reason why we show best and worst results. The average error is around 7 people, and it gives better results when the number people in the room is smaller. This is a good accuracy for our purpose because the correlation with micro-climatic information can be done by creating some occupancy classes, for instance room empty, few occupancy, medium occupancy or room full, and so reducing even more the error rate.

1.5.2 Algorithm Performance

To evaluate the performance of the algorithm we used a different platform than the one presented in Section 1.4.1. Tests has been conducted on the Odroid XU, an ARM equipped developer board produced by Hardkernel company. This platform permits to have more accurate power and performance measurements, than the Samsung Galaxy used to deploy the system, while it embeds a similar class of SoC. In Table 1.1, performance comparison is presented as a function of the scaling size of input image. We used different sized images: 1600×1200, the original size, 800×600, 640×480 and 320×240, to understand the behavior of the algorithm with different inputs.

Finally, we show another comparison to correlate RMSE with performance. We added to this graph the size of input images to give a

Table 1.1: Algorithm Performance comparison

	1600 × 1200	800 × 600	640 × 480	320 × 240
Performance (ms)	660	164	106	27
Image size (KByte)	5,625	1,406	900	450

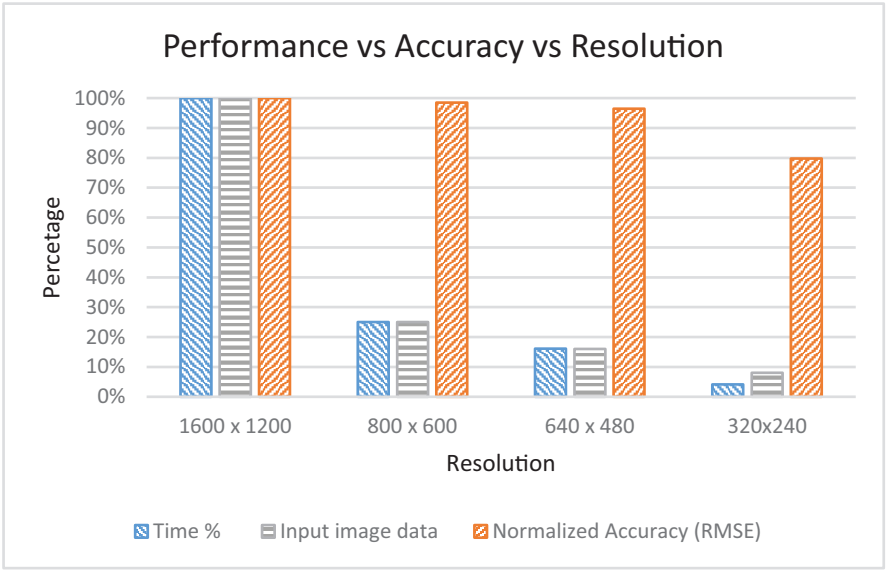


Figure 1.6: Performance, accuracy and resolution comparative chart

quantitative feedback of the differences in amount of elaborated input data. This results are shown in Fig. 1.6.

As we expected the best accuracy is reached with the biggest input image, but what is noteworthy is that scaling down the image of a half and then to VGA size the performance increases of 4.00× for 800×600 px size and then of 6.21× for 640×480 px size, but in both last two cases the accuracy is really similar. Whereas the resolution 320x240 brings a great speedup (24×), but the accuracy of the algorithm in this case drops by 20%. This shows that between the total sized image and the three resized ones, there is a loss of details mandatory to keep the best accuracy for the proposed algorithm. Indeed for timing purpose there is

no need of scaling down the input images, since the sampling time for people counting visual nodes is 10 minutes. This down-sample is related to power consumption discussed in next subsection.

1.5.3 Energy Comparison

The comparison of the energy needed to elaborate the images with the proposed algorithm is presented in Table 1.2. The input of this test is the total number of images taken in a day, 144, and elaborated in batch as a single block of input. The results are compared with four different resolutions.

Table 1.2: Algorithm Energy comparison

	1600 × 1200	800 × 600	640 × 480	320 × 240
Performance (s)	48.56	12.41	7.96	2.22
Energy (J)	229.05	55.90	35.14	12.13

Table 1.2 shows that the effort to produce better results from full size images is much higher and not linear with input data size. Matching results shown in figure 1.6 and table 1.2 we notice that a power/accuracy trade-off could be reached and it is located in 640×480 category. In an energy per sample perspective the elaboration of 1 sample of people counting needs 3.18 J for max resolution and 0.49 J for 640×480. While in a battery perspective, eliminating the system standby, the pure computation on a $4000mAh$ battery will discharge it in about 184 minutes for 1600×1200 and in 192 minutes for 640×480 and the total people counting samples elaborated are about 16751 and 109167 respectively. These numbers shows that we are still far from an autonomous node completely battery-powered, but considering that in one day our test

generates 72 occupancy measurements, the results are considered very interesting, for a quite unexplored application scenario. Indeed further optimization could be done switching ON and OFF internal components and peripherals, improving the global camera nodes power saving.

1.6 Conclusions

In this chapter we presented a real-word application of visual node integrated in a heterogeneous WSN composed of two kinds of nodes: the sensor nodes and the camera nodes. We presented a novel algorithm to detect the occupancy of mostly-static rooms. It elaborates images, instead of video, to reduce the overhead of computation. The on-node implementation permits to reduce the network traffic. The algorithm reaches a good accuracy in mostly-static occupancy contexts. We then explored the trade-off between accuracy and performance on an high-end embedded SoC and we characterized the energy consumption compared to the size of input images. Our results prove that a good trade-off between power consumption and accuracy can be reached for the proposed target to combine occupancy and micro-climatic information.

In next chapter we move from a stationary setting to a wearable one and we explore which are the different requirements and constraints of this different and more challenging setup.

Chapter 2

Gesture Recognition in Ego-Centric Videos

2.1 Overview

In this chapter we move to a more challenging embedded vision paradigm called ego-vision [58]. Ego-centric vision is a paradigm that joins in the same loop humans and wearable devices to augment the subject vision capabilities by automatically processing videos captured with a first-person camera. We present a hand gesture recognition approach for human-machine interfaces designed for a wearable monocular camera. We take into account both static gestures, in which the meaning of the gesture is conveyed by the hand pose, and dynamic gestures, in which the meaning is given by motion too. It should be noted that gestures are somehow personal. In fact, they can vary from individual to individual and even for the same individual between different instances.

In next section we discuss related work, then the gesture recognition algorithm is presented. In Section 2.4 experimental results are presented,

while Section 2.5 concludes the chapter.

2.2 Related Work

The ego-vision scenario has been addressed only recently by the research community and mainly to understand human activities and to recognize hand regions. Pirsiavash *et al.* [59] detected activities of daily living using an approach that involves temporal pyramids and object detectors tuned for objects appearance during interactions and spatial reasoning. Sundaram *et al.* [60] proposed instead to use Dynamic Bayesian Networks to recognize activities from low resolution videos, without performing hand detection and preferring computational inexpensive methods. Fathi *et al.* [61] used a bottom-up segmentation approach to extract hand held objects and trained object-level classifier to recognize objects; furthermore they also proposed an activity detection algorithm based on object state changes [62].

Regarding hand detection, Khan *et al.* in [63] studied color classification for skin segmentation. They pointed out how color-based skin detection has many advantages and potentially high processing speed, invariance against rotation, partial occlusion and pose change. The authors tested Bayesian Networks, Multilayers Perceptrons, AdaBoost, Naive Bayes, RBF Networks and Random Forest. They demonstrated that Random Forest classification obtains the highest F-score among all the other techniques. Fathi *et al.* [61] proposed a different approach to hand detection, based on the assumption that background is static in the world coordinate frame, thus foreground objects are detected as to be the moving regions respect to the background. An initial panorama

of the background is required to discriminate between background and foreground regions: this is achieved by fitting a fundamental matrix to dense optical flow vectors. This approach is shown to be a robust tool for skin detection and hand segmentation in indoor environments, even if it performs poorly with more unconstrained scenarios.

Li *et al.* [64] provide a historical overview of approaches for detecting hands from moving cameras. They define three categories: local appearance-based detection, global appearance-based detection, where a global template of hand is needed, and motion-based detection, which is based on the hypothesis that hands and background have different motion statistics. Motion-based detection approaches require no supervision nor training. On the other hand, these approaches may identify as hand an object manipulated by the user, since it moves together with his hands. In addition they proposed a method with sparse feature selection which was shown to be an illumination-dependent strategy. To solve this issue, they trained a set of Random Forests indexed by a global color histogram, each one reflecting a different illumination condition.

Several approaches to gesture and human action recognition have been proposed. Sanin *et al.* [65] developed a new and more effective spatio-temporal covariance descriptor to classify gestures in conjunction with a boost classifier. Lui *et al.* [66, 67] used tensors and tangent bundle on Grassmann manifolds to classify human actions and hand gestures. Kim *et al.* [68] extended Canonical Correlation Analysis to measure video-to-video similarity to represent and detect actions in video. However, all these approaches are not appropriate for the ego-centric perspective, as they do not take into account any of the specific characteristics of this domain, such as fast camera motion and background cluttering. To our

knowledge, the study of gesture recognition in the ego-centric paradigm has been partially addressed by P. Mistry *et al.* [69]. Their work presents a natural interface to interact with the physical world and embeds a projector to show results of that interaction. However they use colored markers on user's fingers to recognize gestures and they require a back-packed laptop as computational unit. Although our work could seem similar to this last approach, we move a step forward with respect to [69]: we proposed a fully automatic gesture recognition approach based on appearance and motion of the hands. Our approach can deal with background cluttering and camera motion and does not require any markers on fingers.

2.3 Gesture Recognition Algorithm

Gestures can be characterized by both static and dynamic hand movements. Therefore, we consider a video sequence captured by a glass mounted camera, in which a gesture may be performed, and describe it as a collection of dense trajectories extracted around hand regions. When the user's hands appear, feature points are sampled inside and around the hands and tracked during the gesture; then several descriptors are computed inside a spatio-temporal volume aligned with each trajectory to capture its shape, appearance and movement at each frame. We use the following descriptors, according to [70]: Trajectory descriptor, histograms of oriented gradients (HOG), of optical flow (HOF), and motion boundary histograms (MBH). The first one directly captures trajectory shape, while HOG [71] are based on the orientation of image gradient

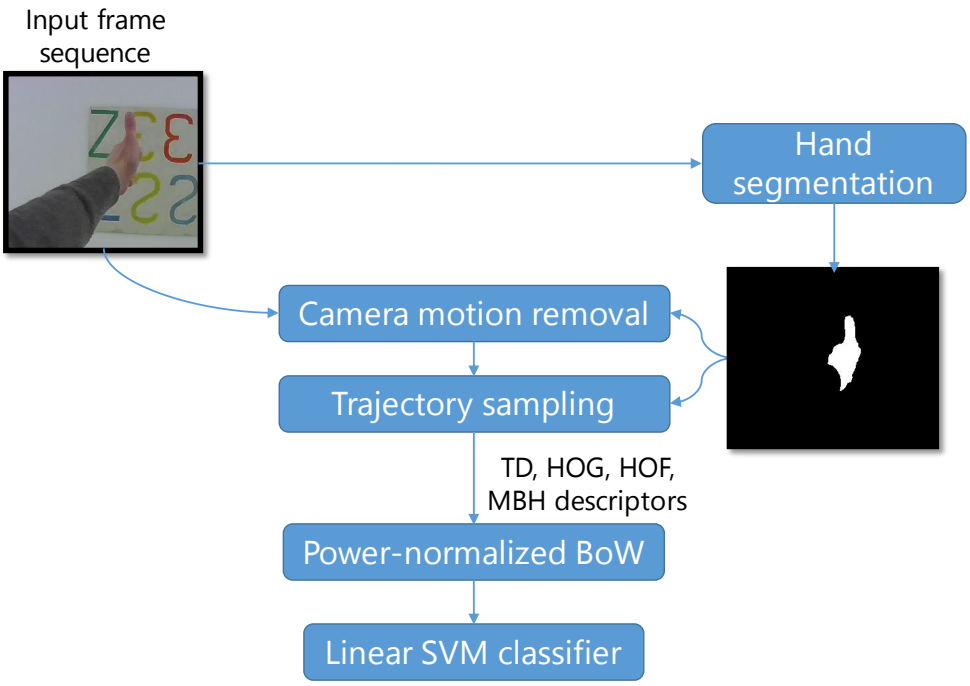


Figure 2.1: Outline of the proposed Gesture Recognition method.

and thus encode the static appearance of the region surrounding the trajectory. HOF and MBH [72] are based on optical flow and are used to capture motion information enforcing the temporal aspect of our method. These descriptors are coded, using the Bag of Words approach and power normalization, to obtain the final feature vectors, which are then classified using a linear SVM classifier. Figure 2.1 provides a more detailed outline of the workflow of the proposed gesture analysis module.

Camera Motion Removal

To estimate the hand motion, it is first necessary to remove the camera motion, which is, semantically, noise. To do so, the homography transform between two consecutive frames is estimated running the RANSAC [73] algorithm on densely sampled features points: SURF [74] features

and sample motion vector are extracted from the Farneback's optical flow [75] to get dense matches between frames. The choice of this particular optical flow algorithm is induced by our preliminary tests, in which Farneback's optical flow showed the best performance when compared to other popular optical flow algorithms, such as TV-L1 [76] and SimpleFlow [77].

In ego-vision, however, it is often the case where camera and hand motions are not consistent, resulting in wrong matches between the frames and degrading the consequent homography estimation. This introduces the need for an additional step based on a totally decoupled feature. We use a hand segmentation mask that allows us to remove the matches belonging to the user's hands, which could have resulted in incorrect trajectories. Computing the homography based only on non-hand keypoints allows to have a motion model consistent with the ego-motion of the camera which can, consequently, be removed.

Gesture Description

After the suppression of camera motion, trajectories can be extracted. Using the previously estimated homography, each frame of the sequence is warped and the Farneback's optical flow between each couple of adjacent frames is recomputed to estimate the motion resulting from the hand movement. Feature points around the hand region are sampled and tracked in a way similar to [70]. We build a spatial pyramid with four layers, such that each layer has half the area of the previous one, and at each spatial scale we apply a threshold on the minimal eigenvalue of the covariance matrix of image derivatives to obtain dense keypoints. We also ensure that keypoints are not duplicated among different spatial

layers, and that a minimum distance between each couple of points is preserved. Each keypoint $P_t = (x_t, y_t)$ is then tracked by the means of median filtering with kernel M in a dense optical flow field $\omega = (u_t, v_t)$:

$$P_{t+1} = (x_{t+1}, y_{t+1}) = (x_t, y_t) + (M * \omega)|_{(\bar{x}_t, \bar{y}_t)} \tag{2.1}$$

where $(\bar{x}_t, \bar{y}_t)$ is the rounded position of P_t. Differently from [70], our trajectories are calculated under the constraint that they lie inside and around the user's hand: at each frame the hand mask is dilated and all keypoints still outside are discarded.

A spatio-temporal volume aligned with each trajectory is then build, as a collection of 32×32 patches around the keypoint. Then, Trajectory descriptor, HOG, HOF and MBH are computed inside the volume. We introduce a difference in how to weight the temporal volume of each component of our feature vector: while HOF and MBH are averaged on five consecutive frames, a single HOG descriptor is computed for each frame. This allows us to describe the changes in the hand pose at a finer temporal granularity. This step results in a variable number of descriptors for each video sequence. To obtain a fixed size descriptor, we exploit the Bag of Words approach training four separate codebooks, one for each descriptor. Each codebook contains K visual words (in the experiments we fix $K = 500$) and is obtained running the k-means algorithm on the training data.

Since the histograms obtained from the Bag of Words in our domain tend to be sparse, they are power normalized to unsparsify the representation, while still allowing for linear classification. To perform

power-normalization [78], the function:

$$f(h_i) = \text{sign}(h_i) \cdot |h_i|^{\frac{1}{2}} \tag{2.2}$$

is applied to each bin h_i in our histograms.

The final descriptor is then obtained by the concatenation of its four power-normalized histograms. Finally, gestures are recognized using a linear SVM 1-vs-1 classifier.

2.3.1 Hand Segmentation

As stated before, a hand segmentation mask is used to distinguish between camera and hand motions, and to prune away all the trajectories that do not belong to the user's hand. In this way, our descriptor captures hands movement and shape as if the camera was fixed, and disregards the noise coming from other moving regions that could be in the scene.

At each frame we extract superpixels using the SLIC algorithm [79], that performs a k-means-based local clustering of pixels in a 5-dimensional space, where color and pixel coordinates are used. Superpixels are then represented with several features: histograms in the HSV and LAB color spaces (that have been proven to be good features for skin representation [63]), Gabor filters and a simple histogram of gradients, to discriminate between objects with a similar color distribution.

Illumination invariance

To deal with different illumination conditions, we cluster the training images running the k-means algorithm on a global HSV histogram. Hence, we train a Random Forest classifier for each cluster. By using a histogram

over all three channels of the HSV color space, each scene cluster encodes both the appearance of the scene and its illumination. Intuitively, this models the fact that hands viewed under similar global appearance will share a similar distribution in the feature space. Given a feature vector $\mathbf{l}$ of a superpixel $\mathbf{s}$ and a global appearance feature $\mathbf{g}$, the posterior distribution of $\mathbf{s}$ is computed by marginalizing over different clusters c:

$$P(\mathbf{s}|\mathbf{l},\mathbf{g}) = \sum_{c=1}^{k} P(\mathbf{s}|\mathbf{l},c)P(c|\mathbf{g}) \tag{2.3}$$

where k is the number of clusters, $P(\mathbf{s}|\mathbf{l},c)$ is the output of the cluster-specific classifier and $P(c|\mathbf{g})$ is a conditional distribution of a cluster c given a global appearance feature $\mathbf{g}$. In test phase, the conditional $P(c|\mathbf{g})$ is approximated using an uniform distribution over the five nearest clusters. It is important to highlight that the optimal number of classifiers depends on the characteristics of the dataset: a training dataset with several different illumination conditions, taken both inside and outside, will need an higher number of classifiers than one taken indoor. In addition, we model the hand appearance not only considering illumination variations, but also including semantic coherence in time and space.

Temporal coherence

To improve the foreground prediction of a pixel in a frame, we replace it with a weighted combination of its previous frames, since past frames should affect the prediction for the current frame.

We define a smoothing filter for a pixel x_t^i from frame t as:

$$\begin{aligned} P(x_t^i = 1) = \sum_{k=0}^{\min(t,d)} w_k (P(x_t^i = 1 | x_{t-k}^i = 1) \cdot \\ \cdot P(x_{t-k}^i = 1 | \mathbf{l}_{t-k}, \mathbf{g}_{t-k}) + P(x_t^i = 1 | x_{t-k}^i = 0) \\ \cdot P(x_{t-k}^i = 0 | \mathbf{l}_{t-k}, \mathbf{g}_{t-k})) \end{aligned} \tag{2.4}$$

where d is the number of past frames used, and $P(x_{t-k}^i = 1 | \mathbf{l}_{t-k}, \mathbf{g}_{t-k})$ is the probability that a pixel in frame $t-k$ is marked as hand part, equal to $P(\mathbf{s} | \mathbf{l}_{t-k}, \mathbf{g}_{t-k})$, being x_t^i part of $\mathbf{s}$. In the same way, $P(x_{t-k}^i = 0 | \mathbf{l}_{t-k}, \mathbf{g}_{t-k})$ is defined as $1 - P(\mathbf{s} | \mathbf{l}_{t-k}, \mathbf{g}_{t-k})$. Last, $P(x_t^i = 1 | x_{t-k}^i = 1)$ and $P(x_t^i = 1 | x_{t-k}^i = 0)$ are prior probabilities estimated from the training set as follows:

$$P(x_t^i = 1 | x_{t-k}^i = 1) = \frac{\#(x_t^i = 1, x_{t-k}^i = 1)}{\#(x_{t-k}^i = 1)}$$

$$P(x_t^i = 1 | x_{t-k}^i = 0) = \frac{\#(x_t^i = 1, x_{t-k}^i = 0)}{\#(x_{t-k}^i = 0)} \tag{2.5}$$

where $\#(x_{t-k}^i = 1)$ and $\#(x_{t-k}^i = 0)$ are the number of times in which x_{t-k}^i belongs or not to a hand region, respectively; $\#(x_t^i = 1, x_{t-k}^i = 1)$ is the number of times that two pixels at the same location in frame t and $t-k$ belong to a hand part; similarly $\#(x_t^i = 1, x_{t-k}^i = 0)$ is the number of times that a pixel in frame t belongs to a hand part and the pixel in the same position in frame $t-k$ does not belong to a hand region. Based on our preliminary experiments we set d equal to three.

Spatial consistency

Given pixels elaborated by the previous steps, we want to exploit spatial consistency to prune away small and isolated pixel groups that are unlikely to be part of hand regions and also aggregate bigger connected pixel groups. For every pixel x, we extract its posterior probability $P(x_i^t)$ and use it as input for the GrabCut algorithm [80]. Each pixel with $P(x_i^t) \geq 0.5$ is marked as foreground, otherwise it's considered as part of background. After the segmentation step, we discard all the small isolated regions that have an area of less than 5% of the frame and we keep only the three largest connected components.

2.4 Experimental Results

To compare the accuracy of the proposed gesture recognition algorithm with existing approaches, we test it on the Cambridge-Gesture database [81], which includes nine hand gesture types performed on a table, under different illumination conditions. To better investigate the effectiveness of the proposed approach in videos taken from the ego-centric perspective and in a museum setting, we also propose a far more realistic and challenging dataset which contains seven gesture classes, performed by five subjects in an interactive exhibition room which functions as a virtual museum. Furthermore, to evaluate the hand segmentation approach, we test it on the publicly available CMU EDSH dataset [64] which consists of three ego-centric videos with indoor and outdoor scenes and large variations of illuminations.

After presenting the accuracy we evaluated the performance of the designed method on an x86 Intel based workstations to investigate the

effort required to bring the algorithm to an integrated embedded solution.

2.4.1 Gesture Recognition

The Cambridge Hand Gesture dataset contains 900 sequences of nine hand gesture classes. Although this dataset does not contain ego-vision videos it is useful to compare our results to recent gesture recognition techniques. In particular, each sequence is recorded with a fixed camera, placed over one hand, and hands perform leftward and rightward movements on a table, with different poses. The whole dataset is divided in five sets, each of them containing image sequences taken under different illumination conditions. The common test protocol, proposed in [81], requires to use the set with normal illumination for training and the remaining sets for testing, thus we use the sequences taken in normal illumination to generate the BoW codebooks and to train the SVM classifier. Then, we perform the test using the remaining sequences.

Table 2.1 shows the recognition rates obtained with our gesture recognition approach, compared with the existing ones, presented in related work section. They are tensor canonical correlation analysis (TCCA) [68], product manifolds (PM) [66], tangent bundles (TB) [67] and spatio-temporal covariance descriptors (Cov3D) [65]. Results show that proposed method outperforms the existing state-of-the-art approaches.

We then propose the Interactive Museum dataset, a gesture recognition dataset taken from the ego-centric perspective in a virtual museum environment. It consists of 700 video sequences, all shot with a wearable camera, in an interactive exhibition room, in which paintings and artworks are projected over a wall, in a virtual museum fashion (see figure 2.2). The camera is placed on the user's head and captures a 800×450,

Method	Set1	Set2	Set3	Set4	Overall
TCCA [68]	0.81	0.81	0.78	0.86	0.82
PM [66]	0.89	0.86	0.89	0.87	0.88
TB [67]	**0.93**	0.88	0.90	0.91	0.91
Cov3D [65]	0.92	**0.94**	0.94	0.93	0.93
Our method	0.92	0.93	**0.97**	**0.95**	**0.94**

Table 2.1: Recognition rates on the Cambridge dataset.

(a) *Dislike* gesture

(b) *Point* gesture

(c) *Slide left to right* gesture

Figure 2.2: Sample gestures from the Interactive Museum dataset.

25 frames per second 24-bit RGB image sequence. In this setting, five different users perform seven hand gestures: *like*, *dislike*, *point*, *ok*, *slide left to right*, *slide right to left* and *take a picture*. Some of them (like the *point*, *ok*, *like* and *dislike* gestures) are statical, others (like the two *slide* gestures) are dynamical. This dataset is very challenging since there is fast camera motion and users have not been trained before recording their gestures, so that each user performs the gestures in a slightly different way, as would happen in a realistic context. We have publicly released our dataset[1].

Since Ego Vision applications are highly interactive, their setup step must be fast (i.e. few positive examples can be acquired). Therefore, to evaluate the proposed gesture recognition approach, we train a 1-vs-1 linear classifier for each user using only two randomly chosen gestures per class as training set. The reported results are the average over 100 independent runs.

In Table 2.2 we show the gesture recognition accuracy for each of the five subjects, and we also compare with the ones obtained without the use of the hand segmentation mask for camera motion removal and trajectories pruning. Results show that our approach is well suited to recognize hand gestures in the ego-centric domain, even using only two positive samples per gesture, and that the use of the segmentation mask can improve recognition accuracy.

2.4.2 Hand Segmentation

The CMU EDSH dataset consists of three ego-centric videos (EDSH1, EDSH2, EDSHK) containing indoor and outdoor scenes where hands are

[1] `http://imagelab.ing.unimore.it/files/ego_virtualmuseum.zip`

User	No segmentation	With segmentation
Subject 1	0.91	**0.95**
Subject 2	0.87	**0.87**
Subject 3	0.92	**0.95**
Subject 4	**0.96**	0.94
Subject 5	0.91	**0.96**
Average	0.91	**0.93**

Table 2.2: Gesture recognition accuracy on the Interactive Museum dataset with and without hand segmentation.

purposefully extended outwards to capture the change in skin color. As this dataset does not contain any gesture annotation, we use it to evaluate only the hand segmentation part.

We validate the techniques that we have proposed for temporal and spatial consistency. In Table 2.3 we compare the performance of the hand segmentation algorithm in terms of F1-measure, firstly using a single Random Forest classifier, and then incrementally adding illumination invariance, the temporal smoothing filter and the spatial consistency technique via the GrabCut algorithm application. Results shows that there is a significant improvement in performance when all the three techniques are used together: illumination invariance increases the performance with respect to the results obtained using only a single random forest classifier, while temporal smoothing and spatial consistency correct incongruities between adjacent frames, prune away small and isolated pixel groups and merge spatially nearby regions, increasing the overall performance.

Then, in Table 2.4 we compare our segmentation method with different techniques: a video stabilization approach based on background

Features	EDSH2	EDSHK
Single RF classifier	0.761	0.829
II	0.789	0.831
II + TS	0.791	0.834
II + TS + SC	**0.852**	**0.901**

Table 2.3: Performance comparison considering Illumination Invariance (II), Temporal Smoothing (TS) and Spatial Consistency (SC).

Method	EDSH2	EDSHK
Hayman and Eklundh [82]	0.211	0.213
Jones and Rehg [83]	0.708	0.787
Li and Kitani [64]	0.835	0.840
Our method	**0.852**	**0.901**

Table 2.4: Hand segmentation comparison with the state-of-the-art.

modeling [82], a single-pixel color method inspired by [83] and the approach proposed in [64] by Li *et al.* , based on a collection of Random Forest classifiers. As can be seen, the single-pixel approach, which basically uses a random regressor trained only using the single pixel LAB values, is still quite effective, even if conceptually simple. Moreover, we observe that the video stabilization approach performs poorly on this dataset, probably because of the large ego-motions these video present. The method proposed by Li *et al.* is the most similar to our approach, nevertheless exploiting temporal and spatial coherence we are able to outperform their results.

2.4.3 Performance Evaluations

In this subsection we give a brief overview of the performance of the algorithm on an Intel based workstation, with a i7-2600 CPU that runs

up to 3.40 GHz.

To evaluate execution time, we divide our algorithm in four modules:

1. *Trajectory sampling* module, which includes trajectories extraction and description

2. *Power-normalized BoW* module, that exploits the Bag of Words approach and power normalization to build the final feature vectors

3. *Classification* module, that performs linear SVM classification

4. *Hand Segmentation* module, that runs our hand segmentation algorithm

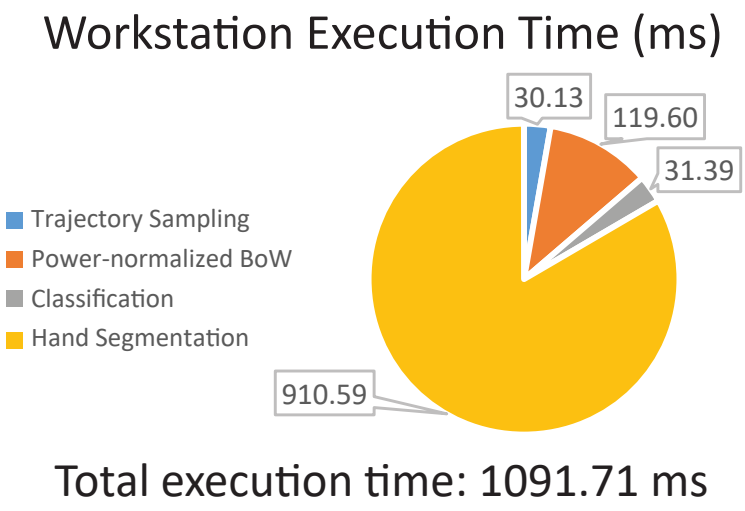


Figure 2.3: Workstation Performance Evaluation on 15 frames trajectories.

Results in Figure 2.3 shows that the total execution time on 15 frames is 1.1 second. This is a good result that means that we reach a near real-time frame rate, resulting in roughly 14 fps. We underline that *Hand*

Segmentation contributes to a greater extent the overall performance compared to the other modules. This consideration will be used in the optimization presented in Chapter 3 to implement our algorithm in a fully integrated embedded system.

2.5 Conclusions

In this chapter we presented a novel gesture recognition algorithm that can deal with static and dynamic gestures and can achieve high accuracy results even when trained with a few positive samples. Our gesture recognition and hand segmentation results outperform the state-of-the-art approaches on Cambridge Hand Gesture and CMU EDSH datasets. Moreover we presented a new Interactive Museum dataset. An accuracy analysis on this dataset has been proposed showing the effectiveness of our approach. Finally we presented a preliminary performance analysis on a workstation class processor.

In next chapter we will show how this CV algorithm can be fully integrated on an embedded system and we further present an architecture in a real-world scenario where this application can be used in cultural heritage to offer a more natural and entertaining way of accessing museum knowledge.

Chapter 3

Fully Integrated Gesture Recognition using Wearable Vision Sensors

3.1 Overview

Museums and cultural sites still lack of an instrument that provides entertainment, instructions and visit customization in an effective natural way. Too often visitors struggle to find the description of the artwork they are looking at and when they finds it, its detail level could be too high or too low for their interests. Moreover, frequently the organization of the exhibition does not reflect the visitors' interests leading them to a pre-ordered path which cultural depth could not be appropriate.

We developed a wearable vision device for museum environments, able to replace the traditional self-service guides and overcoming their limitations and allowing for a more interactive museum experience to all visitors. The aim of our device is to stimulate the visitors to interact

Figure 3.1: Natural interaction with artworks: visitors can get specific content or share information about the observed artwork through simple gestures. Hand segmentation results are highlighted in red and detected gestures are reported in the bottom part of each frame.

with the artwork, reinforcing their real experience, by letting visitors to replicate the gestures (e.g. point out to the part of the painting they're interested in) and behaviors that they would use to ask a guide something about the artwork.

In this Chapter, we show how the algorithm presented in the previous Chapter can be used to recognize user interaction with artworks, and artwork recognition to achieve content-awareness. The proposed solution is based on scalable and distributed wearable devices capable of communicating with each other and with a central server. In particular the connection with the central server allows our wearable devices to grab gestures of past visitors for improving gesture analysis accuracy, to get information and specific content of the observed artwork through the automatic recognition module.

We further demonstrate that our gesture recognition approach can achieve acceptable accuracy results even with a few training samples performed by the visitor, and can benefit from distributed training in which gestures performed by other visitors are exploited. Finally, we present a performance evaluation of our algorithms on an ARM Cortex A15 multi-core platform for wearable devices.

The chapter is structured as follows: in Section 3.2 we give a detailed description of our system, focusing on embedded platform and the system architecture. In Section 3.3 we present a novel dataset taken in a real museum. Then we present accuracy and performance results on an integrated embedded platform.

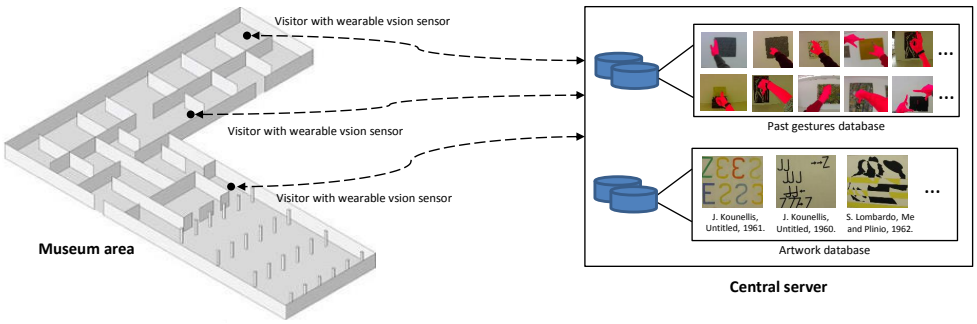


Figure 3.2: Schema of the proposed distributed system. Each wearable vision sensor can communicate with a central server to send captured hand gestures and to retrieve gestures from other users and painting templates for artwork recognition. The central server contains two databases: the gesture database, which includes gestures performed by past visitors, and the artwork database, which contains artwork templates.

3.2 Proposed Architecture

The cultural heritage system consists of a central server and a collection of wearable ego-vision devices, that embed a glass-mounted camera and an Odroid-XU developer board, serving as video-processing and network communication unit. There are several benefits in using such a portable device: the commercial availability and low costs for prototypes evaluation, the computational power and energy efficiency of the big.LITTLE architecture, the possibility of peripheral addition to extend connections and input devices. In particular, the developer board [84] we use embeds the ARM Exynos 5 SoC, that hosts a Quad big.LITTLE ARM processor (Cortex A15 and A7) [85]. To make it a portable demo device a battery pack of 3000 mAh has been added, which guarantees a lasting of 5 to 6 hours(see Figure 3.4).

This wearable device hosts the two main components of our system. The first one is the software that makes it capable of recognizing the gestures performed by its user and can customize itself, learning the way

Figure 3.3: One user interacting with wearable camera.

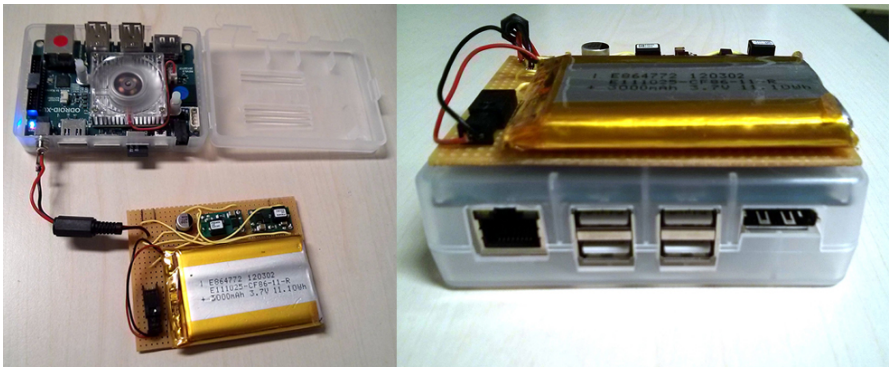


Figure 3.4: The Odroid-XU board with battery pack.

its user reach out for information. Adapting to personal requests is a key aspect in this process, in fact people in different cultures have very different ways of express through gestures. Our method is robust to lighting changes or ego-motion and can learn from a very limited set of examples gathered during a fast setup phase involving the user. The second component of our architecture is the artwork recognition, which allows not only to understand what the user is observing but also to infer the user's position.

The cooperation of ego-vision devices with the central server is two-fold. First, to increase gesture recognition accuracy, wearable devices receive gesture examples performed by past visitors and then send gestures for future users to augment the training set; second, the server also features a database of all the artworks in the museum, which is used for painting recognition and for obtaining detailed text, audio and video content. A schema of the proposed system is presented in Figure 3.2.

3.2.1 Artwork Recognition

In addition to the gesture recognition presented in Chapter 2, the second component of our system is artwork recognition: a matching is established between the framed artwork and its counterpart on the system database. The real-world ego-vision setting we are dealing with makes this task full of challenges: paintings in a museum are often protected by reflective glasses or occluded by other visitors and even by user's hands, requiring a method capable of dealing with these difficulties too.

For this reason, we follow common approaches of object recognition based on interest points and local descriptors [86, 87], that have been proved to be able to capture sufficiently discriminative local elements and are robust to large occlusions.

First of all, SIFT keypoints are extracted from the whole image. The need to proceed with this approach instead of sampling from a detected area derives from the difficulties that arise when trying to detect paintings from a first person perspective. Detection based on shape resulted in high false positive rate, hence we rely on sampling over the whole image. To improve the match quality, we process the matched keypoints using the RANSAC algorithm. The ratio between the remaining matches and the

total number of keypoints is then thresholded, allowing to recognize if the two images refer to the same artwork even in presence of partial occlusions. In addition, to avoid occlusions with user's hands we perform artwork recognition on the frames captured before the recognized gesture using a temporary buffer.

3.3 Experimental Results

3.3.1 Accuracy Evaluation

To test our approach in a real setting, we created a dataset with videos taken in the Maramotti modern art museum, in which paintings, sculptures and *objets d'art* are exposed. As in the previous dataset, the camera is placed on the user's head and captures a 800 $\times$ 450, 25 frames per second image sequence. The Maramotti dataset contains 700 video sequences, recorded by five different persons (some are the same of the Interactive Museum dataset), each performing the same gestures as before in front of different artworks.

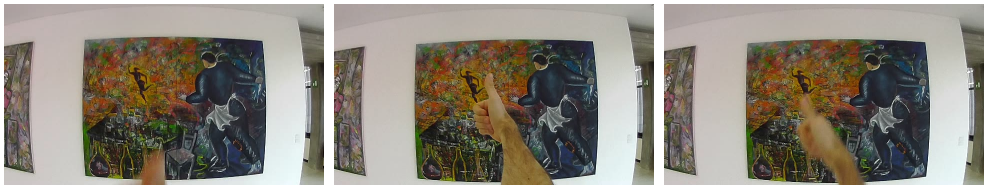
Figure 3.5 show some examples of gestures performed in the dataset. Users perform gestures in front of real artworks inside a museum. This is a realistic and very challenging environment: the illumination changes, other visitors are present and sometimes walk in. In both cases there is significant camera motion, because the camera moves as the users move their heads or arms. It is also important to underline that users have not been trained before recording their gestures, so each user performs the gestures in a slightly different way, as would happen in a realistic context.

In Table 3.1 we show the results of our gesture recognition approach

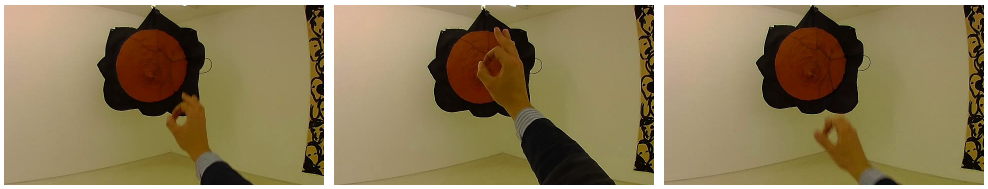
Table 3.1: Gesture recognition accuracy on the Maramotti dataset.

User	Single user's Gestures	Augmented
User A	0.54	**0.65**
User B	0.52	**0.72**
User C	0.68	**0.68**
User F	0.56	**0.79**
User G	0.53	**0.72**
Average	0.57	**0.71**

on the Maramotti dataset. As can be seen, in this case the challenging and real environment causes a drop in accuracy compared to results on Interactive Museum dataset. This is mainly due to the illumination changes, to the presence of other visitors, and to the fact that often the artworks are better illuminated than hands. Since our wearable vision devices is fully connected to a central server, we show how the use of other visitors' gestures can improve the recognition accuracy. In our scenario each visitor coming to the museum performs, in the initial setup phase, two training gestures for each class. These training gestures from past visitors, manually checked, are used to augment the training set, so no erroneous data is accumulated into the model. In particular, in our test "Augmented" (Table 3.1) each ego-vision wearable device uses two randomly chosen gestures performed by its user as training, plus gestures performed by the remaining four users supplied by their devices to the central server. Results show that this distributed approach is effective and leads to a significant improvement in accuracy.



(a) *Like* gesture.



(b) *Ok* gesture, in low light.

(c) *Slide right to left* gesture, while another visitor walks in.

(d) *Take a picture* gesture.

Figure 3.5: Gestures from the Maramotti dataset.

3.3.2 Performance Evaluation

In this section we present our gesture recognition approach performance and optimizations targeting the Interactive Museum Dataset and Maramotti Dataset. They are evaluated on the Hardkernel Odroid-XU board, already introduced in Section 3.2. The tests we further present are performed on the Interactive Museum dataset and the Maramotti dataset. To reach good performance on the Odroid-XU embedded device we applied different optimization techniques. Firstly compiler optimization has

been used to speed-up code execution adding -O3 to compilation flags. Then we used Neon optimized instructions, by including neon library in source code and using these flags at compile time: -mfpu=neon-vfpv4 -mfloat-abi=hard -mtune=cortex-a15 -marm. Several low level "for cycles" have been balanced on different processors using OpenMP parallel regions.

Interactive Museum Dataset

As shown in Table 2.2 the *Hand Segementation* contributes in a small part to the accuracy of the algorithm on this dataset. Thus for the embedded implementation we removed the hand segmentation module that has the worst accuracy/performance contribution to the whole algorithm.

Result shown in Figure 3.6 compares the embedded implementation with the workstation results presented in the previous chapter. Embedded implementation reaches around 19 fps, when the *Hand segmentation* module is disabled. This is a good result that means that we reach a near real-time frame rate. Moreover comparing these results with Figure 3.6 and Table 2.2 it is possible to correlate the accuracy loss, that is around 2%. Hence we trade off a modest accuracy loss for being able to reach near real time performance.

Maramotti Dataset

On the more challenging Maramotti dataset we noticed that the *hand segmentation* contributes in a larger part to the accuracy of the algorithm so we studied a technique to trade-off the accuracy and performance of the solution. We split our algorithm in five main sub-modules (already deeply explained in the previous sections): Hand Segmentation, Camera

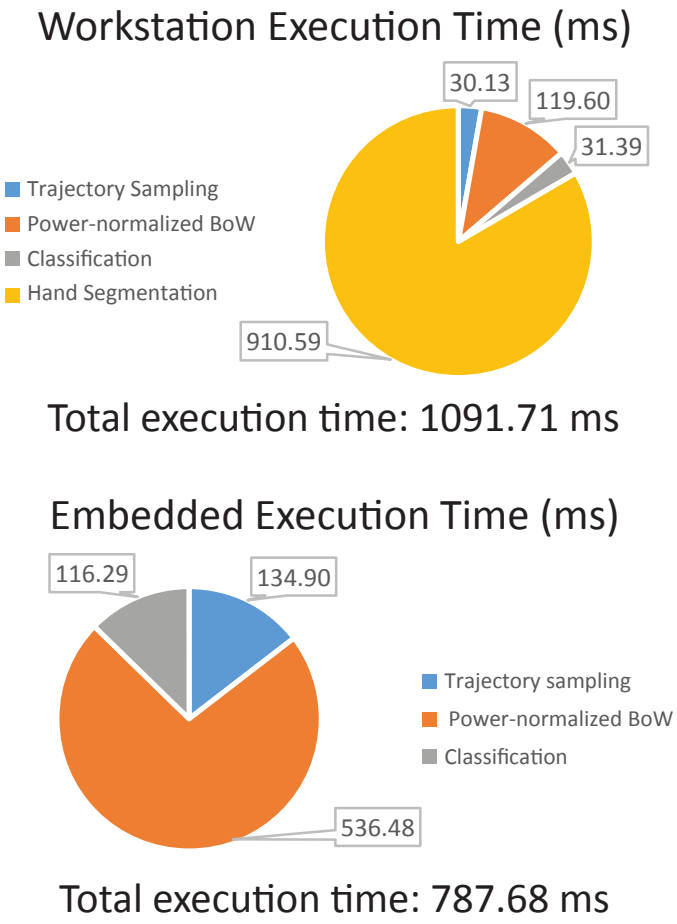


Figure 3.6: Workstation Performance Evaluation on 15 frames trajectories.

motion removal, Trajectory extraction, Trajectory description, Power-normalized BoW and SVM-based Classification. In Figure 3.7 we show the impact of each sub-module, separately, to elaborate 38 frames, that is the average gesture length within the Maramotti dataset. On the bottom part of each column we report the number of times each sub-module is called.

Therefore we introduce a frame step between subsequent elaborations. The idea is to benefit of the hand segmentation not on each frame, but

to introduce a gap between segmentation processing of the video stream and see how this impact on the gesture recognition accuracy. In this case, the hand segmentation mask is computed every s frames. Trajectories and descriptors are still computed using all frames, but new keypoints are sampled only when the hand segmentation mask is available. The choice of higher values of s results in an increasing loss of accuracy, since the gesture is described with less trajectories, but in a decreasing computational power needed to process the input.

Figure 3.8 summarizes the whole gesture recognition algorithm performance and accuracy, applying different hand segmentation frame steps. We evaluated it as an average of the five Maramotti subjects, and the execution step of the Hand Segmentation is evaluated on the average length of the dataset samples (38 frames).

Three lines are shown in the graph: accuracy, performance and the

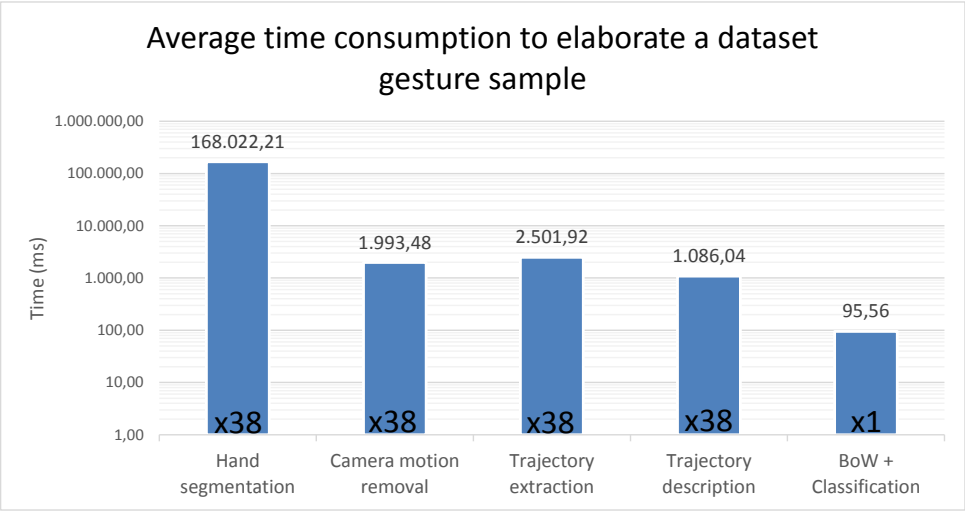


Figure 3.7: Average time of each sub-module to elaborate a gesture sample from the Maramotti dataset.

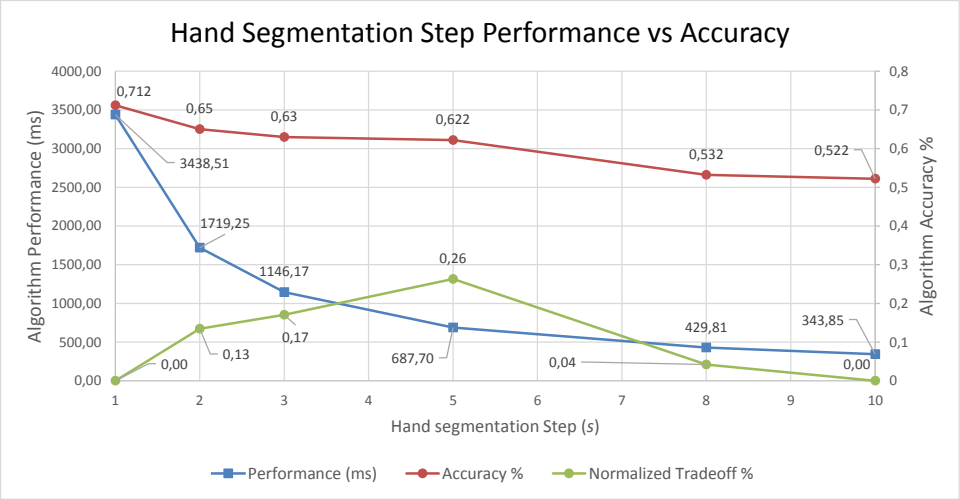


Figure 3.8: Performance-accuracy trade-off of the proposed gesture recognition approach with different Hand Segmentation frame steps.

normalized tradeoff. This last line has been computed as plain multiplication of normalized accuracy by normalized performance. The best normalized tradeoff is given by a step size of 5 frames. The average hands segmentation accuracy decreases of 9% (from 71.2% to 62.2%) in a tradeoff with a speed-up of 5x. This is a good result for performance, because paying a 9% accuracy loss we reduce the execution time from *3438.51 ms* to *687.70 ms*. In Table 3.2 we show a summary of the performances obtained with different step sizes. As can be seen, the best computational performance on Odroid-XU platform is reached when using a step size of 10, and paying an accuracy loss of about 19%. Based on this analysis, we can state that our gesture recognition with hand segmentation is sufficiently accurate for real-life deployment and runs with an acceptable computation performance on ARM-based embedded devices.

Table 3.2: Gesture recognition performance with different step sizes.

Step size	ms per frame	Frame/second	Accuracy
s = 1	3438.51	0.29	71,2 %
s = 5	687.70	1.45	62,2 %
s = 10	343.85	2.91	52,2 %

3.4 Conclusions

We described a novel approach to cultural heritage fruition based on ego-centric vision devices. Our work is motivated by the increasing interest in ego-centric vision and by the growth of the cultural market, which encourages the development of new interfaces to interact with the cultural heritage. We presented a gesture and painting recognition model that can deal with static and dynamic gestures and can benefit from a distributed training. We implemented the presented solution on a fully integrated wearable embedded platform and we ran extensive performance analysis proposing two solutions to Interactive Museum and Maramotti datasets that show the feasibility of the proposed approach.

In next chapter we introduce a low resolution imager for wearable embedded systems of a further low-power class compared to the one presented in this chapter, moving from high-end mobile class to low-end devices.

Chapter 4

Context Change Detection with an Ultra-Low Power Low-Resolution Imager

4.1 Overview

In this chapter we move to the third challenging scenario exploring how, even with very limited resolution, we can obtain context awareness and understand, at least, a change of context in our day-life. We present a context change detector for low-resolution images based on a wearable egocentric camera with ultra-low power consumption. An example of the task that we want to achieve is shown in Fig. 4.1. Low-resolution images can't "see" in the way we usually interpret, as good quality pictures, but can give visual context awareness, that can be exploited for context

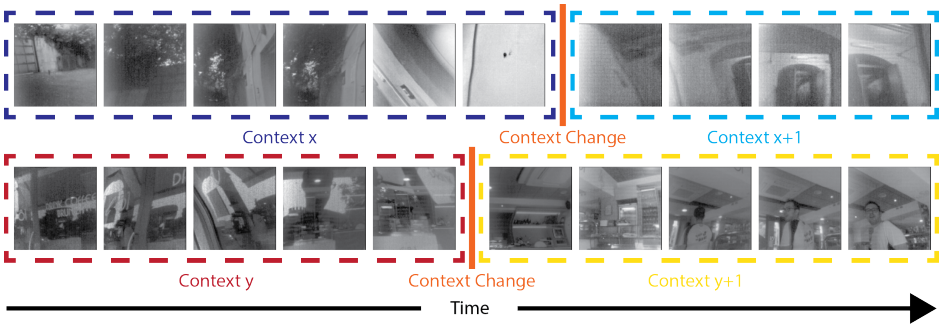


Figure 4.1: We addess the problem of recognizing context changes from low-Resolution images. Figure shows some images taken from the Stonyman Dataset.

change detection. The system is able to collect data 24/7 laying the basis for the long-term analysis of egocentric vision activities. In this context, state of the art context change detection techniques, that are based on results of semantic classifiers, cannot be adopted. Therefore, we propose a novel approach that explores the use of Deep Convolutional Neural Networks on low level resolution images. Experimental results on a new challenging dataset demonstrate that the presented solution is able to detect context changes with good precision.

The chapter is organized as follows. Section 4.2 presents the related work. Section 4.3 gives an overview of the hardware platform and presents the images and the pre-filtering stage. Section 4.4 describes in depth the network architecture, Section 4.5 details the performance and accuracy of our solution, while Section 4.6 concludes the chapter.

4.2 Related Work

Understanding everyday life activities is gaining more and more attention in the research community. This has triggered a number of interesting applications, ranging from health monitoring, memory rehabilitation, lifestyle analysis to security and entertainment [34, 88–90]. These are mainly based on two sources of data: sensor and visual data. Sensor data, such as GPS, light, temperature and acceleration have been extensively used for activity monitoring [91–93]: among others, Kwapisz *et al.* [94] describe how a smartphone can be used to perform activity recognition simply by keeping it in the pocket. Guan *et al.* [95] present a semi-supervised learning algorithm for action understanding based on 40 accelerometers strapped loosely to common trousers. Although sensor data can be easily collected for days, thanks to low energy consumption, its ability to recognize complex activities and the context around the user is low.

On the other hand, computer vision can indeed capture much richer contextual information which has been successfully used to recognize more complex activities [96–98]. Recently, several works that consider vision tasks from the egocentric perspective have been presented. Poleg *et al.* [99] propose a temporal segmentation that identifies 12 different activities (e.g. head motion, sitting, walking etc). Castro *et al.* [100] present an approach based on the combination of a Convolutional Neural Network and a Random Decision Forest; this approach is able to recognize images automatically in 19 activity classes. Ryoo *et al.* [101] suggest a new feature representation for egocentric vision which captures both the entire scene dynamics and the salient local motion observed in

video. There is one notable work which uses adaptive sampling to combine sensor data with camera images [102]. However, these approaches are designed to recognize a limited set of activities and can be useful for specific applications only.

To address this limitation, some unsupervised temporal segmentation and context change detection techniques have been presented, which are capable of splitting an egocentric video into meaningful segments. Lu *et al.* [103] present an approach that discovers the essential moments of a long egocentric video. First, they segment the original video into a series of subshots. Then they represent a short sequence in term of visual objects, that appear within it, using a bank of object detectors. Dimiccoli *et al.* [104] present an approach for context change detection, which combines low-level features and detection of semantic visual concepts (high-level semantic labels are extracted using Imagga's auto-tagging system[1]). By relying on these features, a graph-cut technique is used to integrate agglomerative clustering and an adaptive windowing algorithm [105].

4.3 Egocentric Vision Acquisition System

The egocentric vision acquisition system is a device based on a Texas Instrument MCU which exploits a Stonyman Centeye imager. It is powered by a Li-Ion battery and embeds an energy harvester, that can supply the system while in operation or recharge the battery while the system is in standby. The main advantage of this platform is the ultra-low power

[1]https://imagga.com/solutions/auto-tagging.html

consumption at the expense of image quality. This platform is a development of Infinitime device [106], a wearable bracelet with human body harvesting. In Fig. 4.2 we show the core platform components and a picture of the real device.

The computational unit is an up-to 16 MHz Microcontroller by Texas Instruments, the MSP430FR5969 [107]. This MCU can run in several low power states, turning off unused memories and peripherals, or scaling down the operating frequency. The sensors that this board features are an analog camera, an analog microphone, a temperature sensor and an accelerometer.

4.3.1 Stonyman Imager

The embedded camera sensor, as already mentioned, is a Stonyman sensor by Centeye [108]. Since this is an analog sensor, the first step of the acquisition chain is to sample the images by an Analog to Digital Converter (ADC) and to store them in the system FRAM. Then the platform can store images in an SD card or send them through the NFC to a seconds device (e.g. a smartphone or a tablet).

The analog sensor can capture 112×112 pixel wide grayscale images at

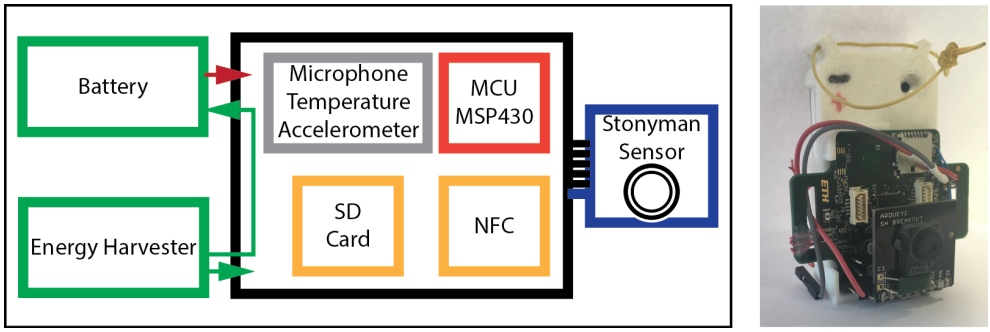


Figure 4.2: Schema of the egocentric vision acquisition system.

up to 2.5 fps, while storing it into SD card. The power consumption of the imager itself is some order of magnitude less than a digital CMOS sensors in the marketplace. In fact, we observed that the power consumption while reading an image is 3.9 mW, while storing an image into SD card takes about 121 mW. In terms of performance, acquiring an image and storing to SD card takes 400 ms. The sending procedure via NFC is less expensive in terms of power budget, as it costs 0.35 mW. In sleep mode the MCU consmes only 0,005 mW. So if we take as example a battery of 1000 mAH, at 1 fps and we store the in the SD Card the device can run for 3/4 days, with a full recharge. Further experiments conducted by Spadaro *et al.* [109] shows that with a kinetic harvester during running activity the harvester can supply enough energy to collect 36 images per minute, while walking activity permits to take 6 images per minute using the NFC to send the image to second device. The ultra-low power consumption showed enables this device to be used as a visual aware sensor.

Images captured by the imager and converted by the ADC are rather noisy, so we pay in some sort in image quality the gain in power consumption. A pre-filtering step is thus required to enhance the image quality. Next section shows the quality of the images and the noise removal technique that we propose.

4.3.2 Images Pre-Processing

Images are sampled by a 12 bit ADC, so a normalization stage is needed before converting them in a 8-bit single channel format. In particular, images sampled from the Stonyman imager are mainly affected by static noise. Therefore, a noise removal stage is carried out with an easy but

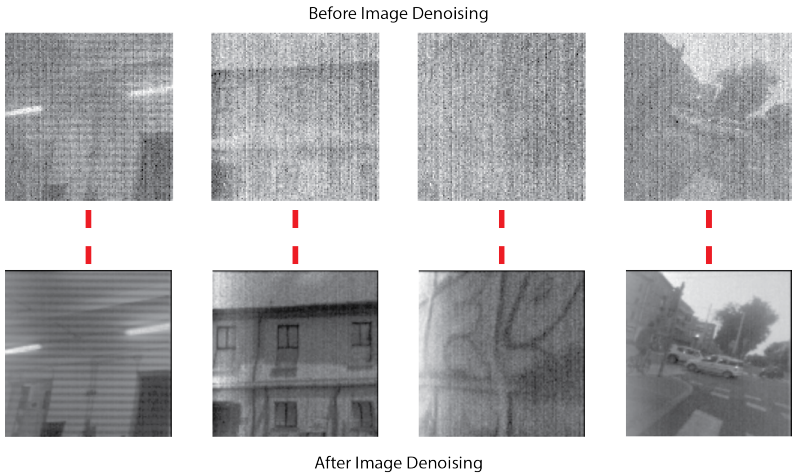


Figure 4.3: Image Denoising results.

effective technique. First a mask is created by averaging several pictures framing a white background in a average light condition. Then the mask is subtracted to the images. This is step helps the temporal segmentation network presented in the next chapter, since the network is based on pretrained weights on images without noise.

In Figure 4.3 we show same samples of images before and after denoising.

4.4 Temporal Segmentation Network

Learning to detect context changes can be addressed as a similarity learning task. In particular, we propose to learn a function $f(x, y)$ that compares an image x to another candidate image y of the same size and returns a high score if the two images depict the same context and a low score otherwise. The function f will be learned from a dataset of videos with labeled change points.

Given their widespread success in computer vision [110–113], we will

use a deep ConvNet as the function f. The architecture of the network resembles that of a Siamese network [114], which is the most used model for addressing similarity learning with ConvNets. Siamese networks apply the same transformation ϕ to both inputs, and then combine their representations using a distance function. Therefore, function ϕ can be considered as an embedding, while the overall network can be seen as a learnable distance computation model.

To train the network, we employ a discriminative approach, by collecting positive and negative pairs. We define positive a pair of images which share the same temporal context, and negative a pair of images sampled from different contexts. At each iteration, we randomly sample a set of pairs $\mathcal{P}$, and minimize the following contrastive loss function:

$$L(\mathbf{w}) = \frac{1}{|\mathcal{P}|} \sum_{(x_i,y_i)\in\mathcal{P}} y_i f(x_i, y_i) + (1 - y_i) \max(0, 1 - f(x_i, y_i)) \qquad (4.1)$$

where $y_i \in \{0, 1\}$ is the ground truth label of each pair. We choose to define the distance function f with respect to the embedding function ϕ through the cosine similarity:

$$f(x, y) = 1 - \frac{\phi(x) \cdot \phi(y)}{\|\phi(x)\| \cdot \|\phi(y)\|} \qquad (4.2)$$

This choice, compared with more popular distance functions for Siamese networks, such as L_1 or L_2, presents a significant advantage. By computing the angle between $\phi(x)$ and $\phi(y)$, and neglecting their magnitudes, it does not force the network to bring its activations into a given numerical range, thus saving training time and avoiding poor local minima.

Table 4.1: Stonyman and Stonyman Quality Datasets sets, number of images and number of context changes (CS).

Set Name	Stonyman D.	Stonyman Quality D.	# of CS
2016-04-06	2734	2143	7
2016-07-05 - 9.00	12104	9257	13
2016-07-06 - 9.00	6256	5566	11
2016-07-07 - 9.00	2056	1544	5
2016-07-07 - 12.00	4367	4043	6
2016-07-08	868	424	3
2016-07-09	876	435	3
Total	29261	23412	48

4.5 Experimental Results

In this section we present the evaluation of our system in terms of accuracy in context change detection. The evaluation has been done by collecting a dataset of images that is described in the next section. In section 4.5.2 we describe the evaluation measures, while in Section 4.5.3 we present accuracy in comparison with two baselines and a state-of-art work.

4.5.1 Stonyman Dataset

To evaluate our results we collected a dataset of 29261 images named "Stonyman Dataset", from the name of the imager. All the images are collected at 1 fps and from a single subject under several days. We define context change any point of the sequence which delimits two temporal segments representing different environments (i.e. we considered as context change going in a shop, enter in the workplace, going off for a pause, etc).

In Table 4.1 we show the sets in which the dataset is divided and the number of images collected per day, while the third column shows the

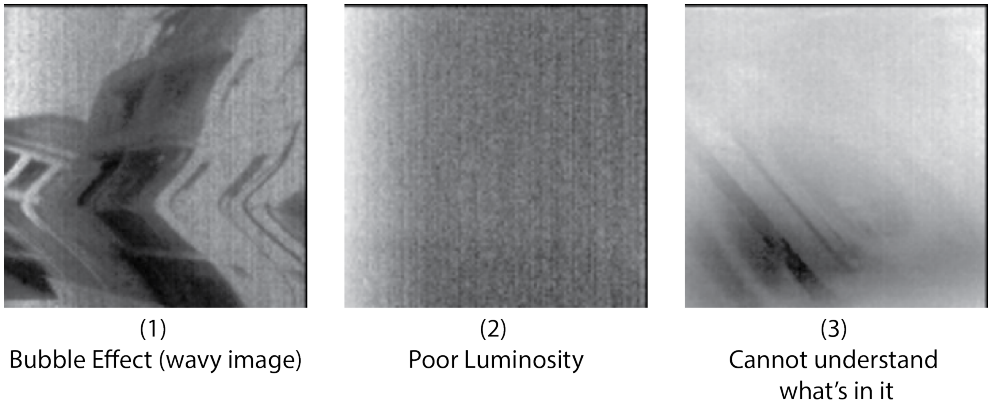


Figure 4.4: Three examples, matching the three criteria used to remove images from Stonyman Dataset to create Stonyman Quality Dataset.

number of images of a subset of the dataset that we called "Stonyman Quality dataset". This is an improved version of the dataset obtained by pruning images with poor quality or that cannot be understood neither by a human expert. In particular, three criteria were considered:

1. Images with bubble effect (wavy images).
2. Images with poor luminosity or completely black.
3. Images where the subject that took the dataset cannot understand what's in it.

In Figure 4.4 an example of each of these defects is shown. This subset has been created to fairly evaluate the results of our system. In case a global shutter is employed by the imager the wavy effect disappears even under low light or fast moving scenes.

4.5.2 Evaluation measures

For the evaluation of context scene detection, the classical precision-recall scheme has been often used, with the important variation of adding a

temporal tolerance factor to detections and ground truth cuts. Therefore, a detection is considered as positive if its distance to nearest ground truth cut is below a certain threshold, otherwise it is considered as a false positive. False negatives are computed by counting ground truth cuts which are further than the same threshold to the nearest detected cut. Formally, given a threshold θ, a set of detected change points $D = \{t_0, t_1, ..., t_n\}$ and the set of ground truth cuts $C = \{t_0^g, t_1^g, ..., t_m^g\}$, true positives, false positives and false negatives are computed as follows:

$$TP = \sum_{i=0}^{n} \max_{j=0}^{m} 1(|t_i - t_j^g| \leq \theta) \quad FP = \sum_{i=0}^{n} 1 - \max_{j=0}^{m} 1(|t_i - t_j^g| \leq \theta)$$
$$FN = \sum_{i=0}^{m} 1 - \max_{j=0}^{n} 1(|t_j - t_i^g| \leq \theta) \tag{4.3}$$

where $1(\cdot)$ is an indication function that returns 1 when the given condition is true, and 0 otherwise. F-Score is then derived from Precision and Recall as usual.

Of course, the major drawback of this measure is the need to set an appropriate tolerance threshold. In our experiments, following previous works in the field [104], we set up a tolerance threshold of 5 frames, which given our frame rate correspond to 5 seconds.

The problem we address can be regarded as a temporal segmentation task, so appropriate measures can be taken from works that addressed temporal segmentation in other scenarios. One of them is surely scene detection, in which the objective is to temporally segment a broadcast video in semantically meaningful parts. In this settng, a measure based on intersection over union has been recently proposed [115]. Here, each

temporal segment is represented as a closed interval, where the left bound of the interval is the starting frame, and the right bound is the ending frame of the sequence. The intersection over union of two segments a and b, $\mathrm{IoU}(a, b)$, is written as

$$\mathrm{IoU}(a, b) = \frac{a \cap b}{a \cup b} \tag{4.4}$$

A segmentation of a video can be seen as a set of non-overlapping sequences, whose union is the set of frames of the video. By exploiting this relation, [115] defines the intersection over union of two segmentations C and D as:

$$\overline{\mathrm{IoU}}(C, D) = \frac{1}{2}\left(\frac{1}{\#C}\sum_{a\in C}\max_{b\in D}\mathrm{IoU}(a, b) + \frac{1}{\#D}\sum_{b\in D}\max_{a\in C}\mathrm{IoU}(a, b)\right) \tag{4.5}$$

It is easy to see that Eq. 4.5 computes, for each ground-truth segment, the maximum intersection over union with the detected segments. Then, the same is done for detected segments against ground-truth ones, and the two quantities are averaged.

4.5.3 Results

To quantitatively evaluate the difficulty of dealing with low resolution images, we first present two baseline experiments. They both use Histogram of Oriented Gradients [116] (HOG) as descriptors, and hierarchical agglomerative clustering with euclidean distance to group images in contexts. The choice of this method provides a similarity measure of images of the same scene vs images of different scenes.

In the former baseline test (named CT1, Clustering Test 1), we fix

the number of clusters to eight, which is the number of unique contexts that we have in our dataset: biking, car, home, office, walking, stairs, supermarket/shop, outdoor. The idea behind this experiment is to test the ability of a popular hand-crafted descriptor to distinguish between different contexts and places. HOG are extracted separately from each image and then descriptors are clustered in eight clusters.

In the latter test (named CT2, Clustering Test 2), instead, agglomerative clustering is applied with a different methodology, which resembles that of a Siamese network. Images are elaborated in subsequent couples from the beginning to end of the dataset. From each couple of images we extract HOG features, and compute the element-wise L_1 distance on feature vectors. We thus get a feature vector for each couple, having the same dimensionality of the HOG descriptor. The resulting features are then given as input to the agglomerative clustering, but instead of looking for eight clusters as the previous baseline test, we fix the number of clusters to two (similar and dissimilar pairs).

In Figure 4.5 and 4.6 we present the accuracy measured respectively with F-Score and IoU on CT1 and CT2. We tested two different settings for HOG features extraction. For both we used a window size of 112×112, block size of 56×56 and block stride of 28×28, and tested two different cell sizes: 28×28 and 56×56. We selected these two settings after conducting a grid search on a subset of the dataset, and picked the top two feature sizes in accuracy.

As it can be seen from the two charts, F-Score and IoU values are very low, thus revealing that hand-crafted features are not well suitable for low-resolution noisy images. The best accuracy in terms of F-Score is achieved with CT1, since the solution of clustering into eight classes is a

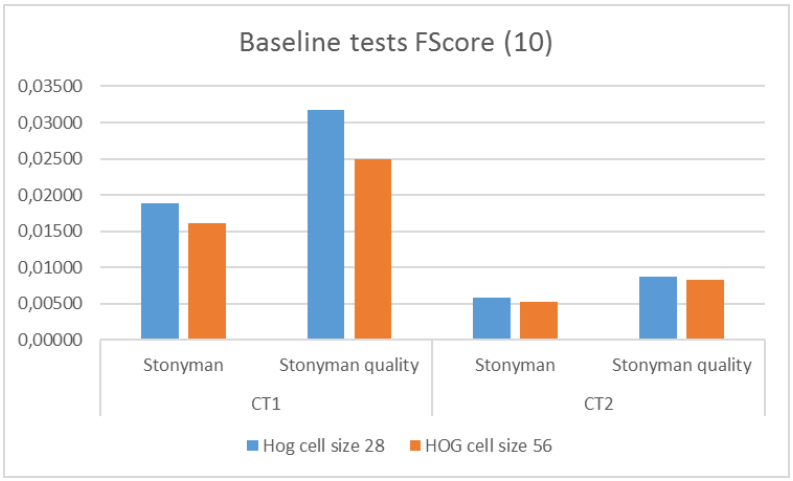


Figure 4.5: CT1 and CT2 baselines in terms of F-Score

more easy task, and we see a slight improvement with Stonyman Quality with respect to the entire dataset. In Figure 4.6 the same results are evaluated in terms of IoU. All settings results in similar values of IoU, and this is due to the completely different nature of the two performance measures.

Moving to the proposed approach, we employed the pre-trained 16 layers model from VGG [110] as the embedding function ϕ, since it is well known for its state-of-the-art performances on image classifications tasks, while still being a simple and lightweight model for modern GPUs. The overall network is then trained end-to-end using Stochastic Gradient Descent with learning rate 0.001 and batches of 20 couples.

In Table 4.2 we present the results of our system on the Stonyman and Stonyman Quality datasets. The performances are reported in terms of F-Score and IoU for each set. Notice that Stoneyman Quality compared to Stonyman produce 0.2 improvement in F-Score and 0.1 improvement in IoU, we attribute this behavior mostly to wavy images that are removed in Stonyman Quality. These distort images produces an altered feature,

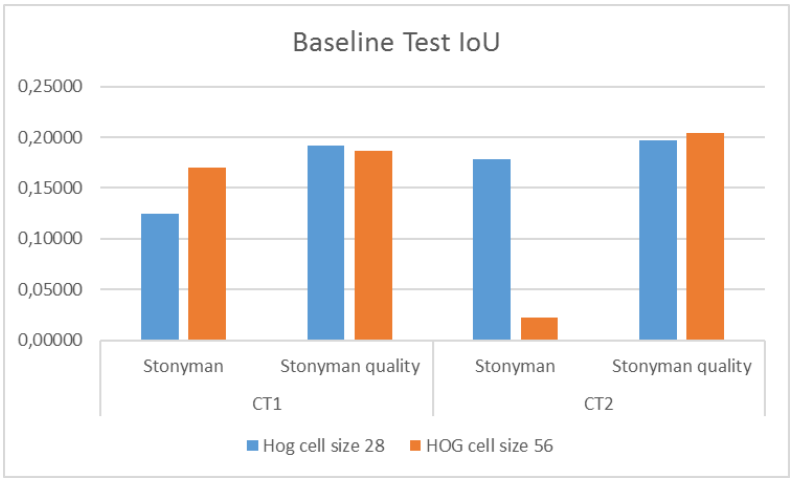


Figure 4.6: CT1 and CT2 baselines in terms of IoU

Table 4.2: F-Score and IoU results of our system on Stonyman and Stonyman Quality datasets

	Stonyman D.		Stonyman Quality D.	
	F-Score	IoU	F-Score	IoU
2016-04-06	0.571	0.655	0.667	0.608
2016-07-05	0.216	0.411	0.357	0.539
2016-07-06	0.105	0.590	0.286	0.375
2016-07-07 - 9.00	0.133	0.397	0.625	0.791
2016-07-07 - 12.00	0.217	0.387	0.500	0.712
2016-07-08	0.143	0.618	0.400	0.552
2016-07-09	0.193	0.346	0.267	0.520
Average	0.226	0.486	0.443	0.585

that make the problem more challenging.

Table 4.3 present a comparison of the two baselines (CT1 and CT2) and our system. We can observe that the techniques based on scene clustering achieve low performance. Whereas our system obtains promising results in both scenarios. We could not compare our solution with a state of the art Scene Clustering System called SR-Clustering [104], because, as mentioned before, a key element of their technique is the extensively

Table 4.3: Comparison results between the proposed solution and the two baselines (CT1 and CT2) on Stonyman and Stonyman Quality datasets.

	Stonyman D.		Stonyman Quality D.	
	F-Score	IoU	F-Score	IoU
CT1	0.019	0.170	0.032	0.192
CT2	0.006	0.179	0.009	0.204
Our System	**0.226**	**0.486**	**0.443**	**0.585**

usage of high-level semantic classifiers, which don't work with our low-resolution snapshots. This is clearly shown in Figure 4.7, in which we present some examples of predictions obtained on our low-resolution images by the classifiers adopted in [104] comparated to the corresponding narrative images.

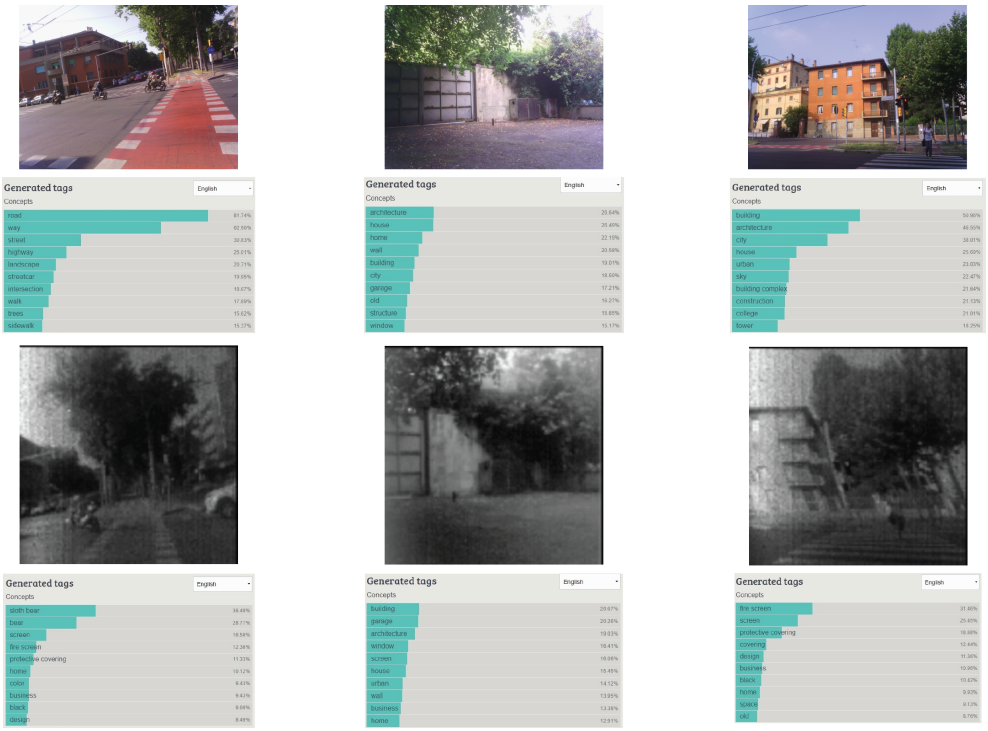


Figure 4.7: This figure shows Imagga predicted tags on the same images shot with Stonyman (Grayscale) and Narrative (Color)

Therefore even if our system cannot exploit an high-level semantic

Table 4.4: Performnace results of our system on EDUB-Seg

	F-Score	IoU
Subject1_1	0.563	0.494
Subject1_2	0.545	0.536
Subject2_1	0.448	0.466
Subject2_2	0.500	0.473
Subject3_1	0.500	0.418
Subject3_2	0.400	0.574
Subject4_1	0.476	0.546
Subject4_2	0.774	0.560
Average	0.521	0.510

classifier, we tested it on the reference dataset of SR-Clustering to show that results on color high quality images are in-line with the Stonyman Quality low-resolution images.

The SR-Clustering work proposes a dataset called EDUB-Seg which is composed of two sets: EDUB-Seg Set 1 and EDUB-Seg Set 2. The only publicly available one is EDUB-Seg Set 1. This dataset is composed 4912 color images (512×512 pixels) collected by 4 subjects with a Narrative Clip camera [117] at 2 fpm. In Table 4.4 is shown our results on this dataset. We trained the network with the technique leave-one-out: for each subset the network is trained on all the other subsets. The results shows an improvement in F-Score compared to Stoneyman Quality dataset, while on IoU there is a slight loss. This shows that in this dataset the low framerate is balanced by the quality of the images.

Table 4.5: Accuracy of our system and SR-Clustering in EDUB-Seg Set 1 Dataset

	F-Score
SR-Clustering	0.69
Our System	0.521

Lastly in Table 4.5 we report for the reader the results of SR-Clustering

on EDUB-Seg set 1 and the average F-Score that we achieve with our system.

4.6 Conclusions

In this chapter we proposed a context change detection system. First, we presented an egocentric vision device with ultra-low power consumption that can capture images round the clock. Then, we suggested a similarity learning approach, based on Siamese ConvNets, that is able to deal with grayscale low-resolution snapshots. We finally ran extensive experiments in real scenarios, showing the robustness and efficacy of the proposed method with respect to related approaches.

The possibility to integrate and deploy this kind of applications on a low-end platform poses two clear challenges. The first and more obvious one is in terms of computational resources that the studied solution needs. Such an MCU presented in this chapter and employed as ego-centric embedded vision device has a reduced amount of computational power and memory available. Our design and the use cases proposed make it possible to find a tradeoff by reducing the sample rate of images. Therefore in terms of performance and memory footprint some techniques can be studied and employed in terms of hardware choice [118] and software optimizations [119]. The deployment in a real-world scenario of such an application on a MCU platform rises, moreover, concerns in terms of security, and this is the second challenge that must be addressed to bring this class of systems to an integrated solution that can be flexibly programmed by third-party developers. The context change detection algorithm is one of the possible future usages of this vision sensor, but

there exist several others: face recognition, video summarization, life logging, etc. Thus, to provide flexibility and security to third-party software development and deployment, we designed and developed a framework for MCUs device class as the one presented in this chapter. In the next chapter we deal with this second challenge, explaining the problem and presenting our proposed solution.

Chapter 5

Lightweight Virtualization on MPU Enabled Microcontrollers

5.1 Overview

An important aspect in the deployment of applications presented in chapter 4 is the flexibility and the security of their execution. The virtualization of the hardware resources becomes necessary to execute securely third-party software and different applications with well-controlled interference. Then, the capability to remotely download new parts of code, to link dynamically the binary and to execute runtime within the main application permit to support more flexible software updates, avoiding moreover down-time and reboot. However, if this technology is well known and available in operating systems for high-end embedded systems (e.g. Linux on ARM Cortex-A microprocessors), providing mechanisms

for dynamic linking in low-resource microcontroller based embedded platforms, such as ARM Cortex-M class, is still a challenge, and only few and limited solutions have been proposed so far.

In this chapter we present a framework which provides a lightweight virtualization of the IO and platform peripherals and permits the dynamic loading of new user code. The aim of this work is to support critical isolation features typical of virtualization-ready CPUs on low-cost low-power microcontrollers with no MMU (Memory Management Unit), IOMMU or dedicated instruction extensions. Our approach only leverages the Memory Protection Unit (MPU), which is generally available in all ARM Cortex-M3 and Cortex-M4 microcontrollers.

The chapter is organized as follows. Section 5.2 gives an overview of the work related to our IoT Lightweight Virtualization software infrastructure, Section 5.3 describes in depth the framework architecture and provides all technical details of this solution, Section 5.4 details our performance and memory footprint, while Section 5.5 concludes the chapter.

5.2 Related Work

Virtualization support for embedded systems based on high-end CPUs, such as the ARM Cortex-A series, has been extensively explored in the academic literature and has reached industrial maturity [120]. This class of devices exploits the hardware extensions to provide hardware abstraction and protection of critical resources. Recent Cortex-A CPUs feature native virtualization support like MMU and IOMMU address translation, interrupt virtualization, TrustZones [121, 122], etc. Cortex-M MCUs do

not come with any of those hardware extensions. Furthermore, available memory and computational resources are much more limited. Our work and the related works surveyed below deal with Cortex-M3 and Cortex-M4 class of devices, where virtualization is not a mature technology and several compromises with respect to full hardware-supported virtualization have to be made.

Abstract Virtual Machines and Interpreters

One of the most common approaches for virtualization on MCUs is based on interpreter-based virtual machines, which have been originally conceived with the main purpose of creating high-level easy-to-use languages and run-times at a higher abstraction level than the traditional C language. Python [123, 124], Java [125, 126], Javascript [127], Lua [128] are all lightweight multi-paradigm scripting languages employed in Virtual Machines for embedded systems. Their main benefit is the cross-platform support. They are interpreted by a native virtual machine loaded on the microcontroller, thus they introduce high overhead in term of latency of access to the resources in comparison to virtualization layers written in native code, but they are designed for easy software application develoment and to meet the increasing demand of fast run time customization, without the need of complex or dedicated compiling toolchains. Such a kind of virtualization, usually, is focused on improving portability, extensibility, ease-of-use in development and protection but lacks performance, multiple user level accesses and low-level hardware control. Only the exposed high level resources can be leveraged by the user.

Bogliolo *et al.* [129] presented *Virtual Sense*, a sensor node which executes java-compatible virtual machine called *Darjeeling VM* [126] on

top of Contiki OS [130]. This work is close to ours in the emphasis on supporting resource allocation and protection for multiple independent user tasks on the MCU. However this solution, besides the overhead introduced by the interpreter, is oriented to share only network stack between Darjeeling VM tasks, while our work is general to all peripherals.

Just In Time/Ahead of Time Compilation

A well-explored approach to reduce the run-time overhead of VM interpeters is Just in Time or Ahead of Time Compilation. Micropython [123] developers, for example, introduced in their platform the concept of *decorator* to emit ARM native *opcode* and to use native C types, but not all native C types are supported and the implementation of this optimization is platform dependent. A solution can be to extend with C wrapped functions called from python, but there are drawbacks: marshaling and unmarshaling of data is very expensive in terms of computational resources and with this solution the programmer loses the low level abstraction. In comparison, using our solution, the developer implements C functions which will be executed in user level tasks. In general these approaches require a higher memory footprint to host the just-in-time or ahead-of-time compile process and do not achieve the performance of native code execution. Furthermore, they are difficult to use in contexts where real-time constraints cannot tolerate the jitter introduced by on-line compilation.

Native Implementations

Native virtualization is the closest to hardware and extremely desirable for resource and performance-limited devices. This technique usually relies on the use of MPU that is the only hardware unit available for security in low-end systems.

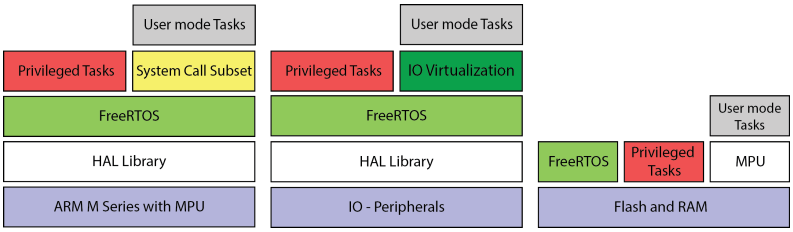


Figure 5.1: Hardware, IO and Memories layers.

Bhatti *et al.* [131] presented a complete operating system designed for WSN (Wireless Sensor Network) and optimized to simultaneous execution of threads which can be loaded dynamically. Their work relies on *Mantis OS*, a custom operating system. They work targets micro sensor nodes with 4KB of RAM. They support dynamic reprogramming at runtime of variables/parameters, while to add new functionalities, differently from our work, a reset is needed. Moreover they do not explicitly address security and protection.

To the best of our knowledge we find only one very recent work that addresses the problem in a broad and general sense, similarly to our solution. Andersen *et al.* [132] presented an embedded platform that relies on TinyOS. They use a mixed paradigm that permits to have Lua VM but the computational intensive part of code can be written in native C. To address security they use a task receiving event based system calls, to separate kernel to user space tasks. Our work differentiates from the latter by permitting to have both system call support and event based peripheral virtualization. Moreover Andersen *et al.* do not provide any information on the performance of the event based system call paradigm.

5.3 Software Architecture

In this section we present all the software layers in our runtime system, focusing on software protection. Figure 5.1 shows the layer stacking from three viewpoints, first from a hardware point of view, then from address space access, divided in IO and Flash/RAM. We divided core hardware from peripherals in two different stacks to underline that the OS can expose system calls to access to the core hardware resources, while the Virtual IO Layer is designed to access the peripherals. The last stack shows that the access to memories is direct for privileged tasks, while the access from usermode tasks is strictly regulated by MPU. Two different kinds of tasks are defined: privileged tasks and usermode tasks, which will be discussed in next section.

Another important layer depicted in Figure 5.1 is FreeRTOS [133], a well known Real Time Operating System for a broad range of Embedded Systems from 8 to 32bit, including low power and ultra-low power MCUs. We implemented our framework on an STM32F4 based platform, and even if some details in the following description are related to this specific microcontroller, our framework can be easily extended to be platform independent.

In Sections 5.3.1 and 5.3.2 we focus on the first and third stack, namely on exploiting the MPU and providing Safety Extensions, while in Section 5.3.3 we discuss the second stack.

5.3.1 Real Time OS

The main reason for using FreeRTOS is its versatility: it is open source with modified GPL license, many MCUs are supported and the code is

maintained and upgraded often by Real Time Engineers Ltd. Moreover it is modular and there are some extensions available (e.g. MPU extension), which can be added to the core release. The open source nature makes possible to extend it. It has moreover a small memory footprint and sources consist of a small number of files. The scheduler supports real-time operation, both time-triggered by a configurable system tick and with support for priorities with preemption.

5.3.2 FreeRTOS Additions

To strengthen the security of the system, the FreeRTOS MPU module has been integrated to enable the usage of the Memory Protection Unit available on the microcontroller and to activate the two levels of privileges for the tasks execution. However, the original module is an experimental release, because of some limitations that we addressed in our work:

1. It does not have a proper way to access system resources. It provides only one system call. This system call raises the privileges of the caller from usermode to privileged, executes the call and then sets the privileges back to user space. This behavior has sufficient protection in an environment where a single developer wants to keep separation between tasks, i.e. the case where a single company develops all the firmware. While in the case we want to give to a third-party user the capability to develop his own code, the knowledge of the existence of this backdoor is really dangerous for protection.

2. The exploitation of the MPU is static. The protection sections of the MPU are not reconfigurable at run-time by privileged tasks

with an API.

3. The task termination is not correctly handled. When a usermode task raises an MPU trap the exception ends the system execution. Hence it would be extremely easy to create denial of service attacks.

In next sub-sections we describe our proposed solutions to these limitations. This solution has been designed and implemented.

MPU Extension

As already stated, this module permits to grant different access privileges on a task-by-task basis. For each task the MPU settings are stored in the task descriptor, called Task Control Block (TCB) in FreeRTOS. When a task is created, it can be started with one out of two levels of privileges:

1. Privileged Tasks (similar to Linux Kernel Mode execution). The task executes with permission granted to access all system resources, memories and peripherals.

2. Usermode Tasks (similar to Linux Usermode, also called unprivileged tasks). The task is executed in more restrictive environment and has access only to a limited subset of memory and IO addresses.

STM32 Cortex-M4 has eight configurable MPU regions. When activated, the protection policy is white-list based for usermode tasks. To access to a specific position in the address space the task should have a grant by one MPU region. For privileged tasks the protection policy is black-list based. The privileges on an MPU region can be: NONE, READONLY AND READWRITE. In FreeRTOS these MPU regions are configured as follows:

Region 0 *FLASH protection*

Protects whole FLASH providing read-only privileges to both privileged and usermode tasks.

Region 1 *OS FLASH protection*

Protects from accesses by usermode tasks to the OS code in FLASH

Region 2 *OS RAM access*

Provides permission to privileged task to access the OS structures stored in RAM

Region 3 *Peripheral access*

Used to enable or disable the access to peripherals.

Region 4 *Task Stack access*

Used to give access to tasks own stack.

Region 5-7 *Not used*

These three regions are not used by FreeRTOS MPU module, thus they are available for developer purposes.

In Table 5.1, we show a list of MPU configurations used in our solution. There is no access to peripherals granted to usermode tasks. The access is allowed only through the IO Virtualization Architecture.

One of the main constraints of the FreeRTOS MPU module is that it permits to configure the last regions (from 5 to 7) at compile time only. Thus, we implemented a specific software module to reconfigure these regions at run-time for each task. This is done for the following reasons:

1. Access to Virtual IO Layer (deeply explained in Subsection 5.3.3) can be restricted by an MPU Region and must be asked by a task.

Table 5.1: Default MPU region setting in FreeRTOS

Privileged Perm.	Usermode Perm.	Region Desc.
READ ONLY	READ ONLY	all Flash Protection
READ ONLY	NONE	OS Code Segment in FLASH
READ WRITE	NONE	OS RAM Protection
READ WRITE	NONE	Peripherals
READ WRITE	READ WRITE	Task Stack
NOT USED	NOT USED	User configurable
NOT USED	NOT USED	User configurable
NOT USED	NOT USED	User configurable

This makes the Virtual IO Layer aware about the number of tasks that are using it.

2. Access to heap or other memory regions can be granted at run-time. This is open to several future applications.

Safety Extensions

As previously stated, the single system call paradigm is not safe. The *raise_privilege* system call has been removed and replaced by more specific system calls. For example to grant access to FreeRTOS Queues and Direct Task Notification, the following list of system calls are added:

- MPU_xTaskGenericNotify: Direct task notification Notify function
- MPU_QueueReceive: Receive a message on a queue
- MPU_xGetCurrentTaskHandle: Get the current task handle
- IO_Layer_REGISTER: Registration to Virtual IO Layer

Graceful Task Termination - Killer Task

FreeRTOS does not provide task termination. Thus, when an unprivileged task tries to access a memory address without permission a *trap* is generated from the MPU and the OS ends its execution in an endless loop. This is not acceptable if we want to keep all other tasks and OS in execution. The desired behavior is that the task causing the *trap*, is aborted while the system continues its execution. Thus a memory trap handler and a specific task, called *Killer Task*, have been created to manage the termination of the task that raised the *trap*. The *Killer Task* is a privileged task created at boot time and it is in *sleep* state, when the MCU is in normal usage. When a *trap* occurs the task is activated. The *Killer Task* gets the task handles of the task that generated the *trap* and removes it from the scheduler execution queue. Then it resumes the scheduler execution and goes back into sleep, waiting for the next *trap*.

5.3.3 IO Virtualization Architecture

In a software protection perspective, the MPU enables the OS to keep the control on the usermode tasks. Thus, with the MPU all usermode tasks cannot tamper the whole system. On the other hand, if we want to enable a third party software developer to access only a small subset of peripherals, a fine grain control on address space must be implemented. Usually in a MCU all peripherals addresses are grouped from a starting to an ending address. However, if we want to provide fine grain access to a subset of them, three free MPU regions are really limiting. Moreover there are other two limitations: one is that the minimum area for an MPU region is usually 32 Bytes (i.e. on STM32f4) that is usually larger

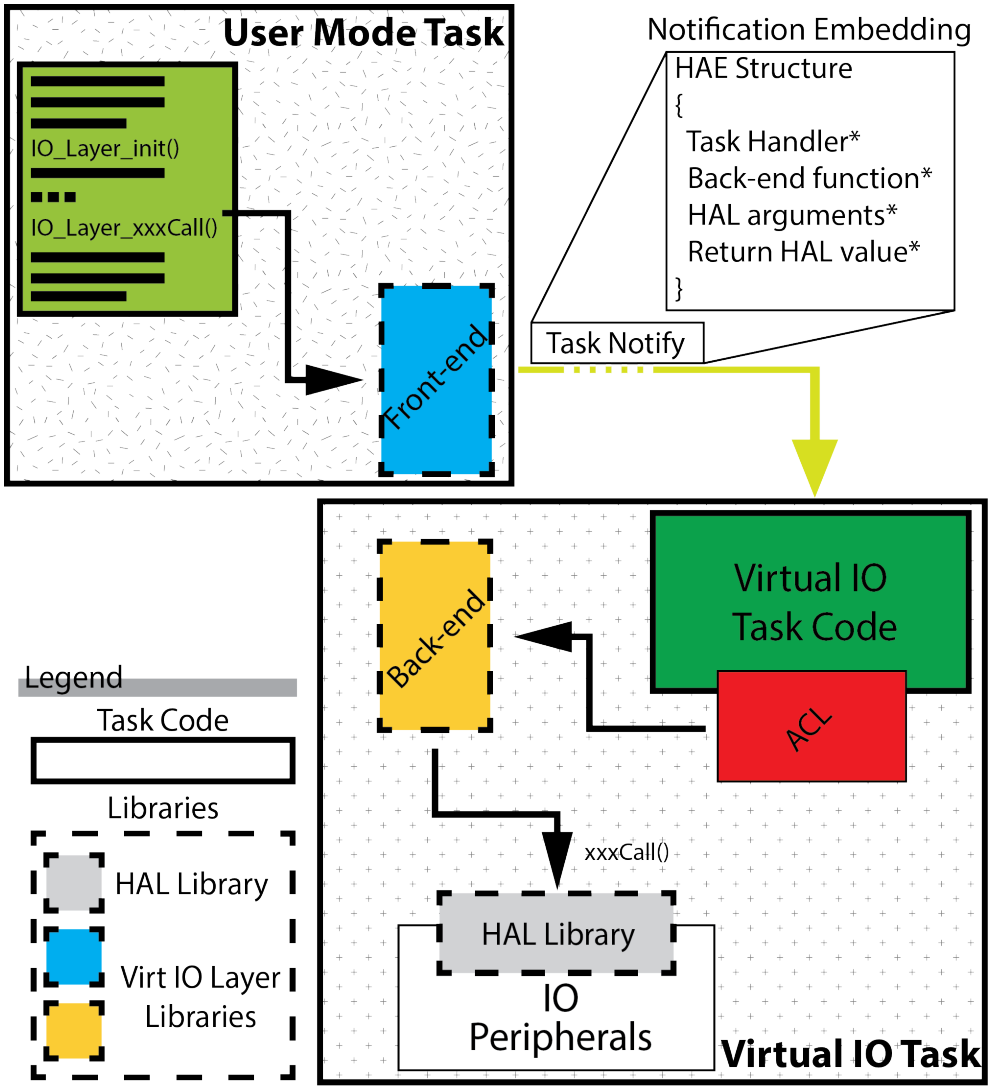


Figure 5.2: IO Virtualization High Level Architecture

than the register pool of a peripheral. The other is that register set of several peripherals consists of both control registers, and reading/writing ports, at subsequent memory positions. Thus it is not possible to grant the access to a read-only register and denying the permission to a contiguous configuration register. The virtualization layer addresses these limitations.

The Virtual IO Layer architecture consists of two main components: (1) a task named *Virtual IO Task* and (2) a library named *Virtual IO Library*. The *Virtual IO Task* is a FreeRTOS task that handles all the IO calls from the usermode tasks to the peripherals. The *Virtual IO Library* contains the *front-end calls*, called from the usermode tasks and forwarded transparently to the *Virtual IO Task*, and the *back-end calls* invoked by the *Virtual IO Task* to access to the peripherals through the HAL Library. As shown in Figure 5.2 the *Virtual IO Task* acts as a task-in-the-middle that receives all calls from usermode tasks that attempt to access to the peripherals, checks the permissions and forwards the requests through the HAL library.

Virtual IO Library

The library consists of two subsets: a front-end functions subset and the relative back-end functions subset.

When a usermode task wants to access peripherals, it needs to subscribe to the Virtual IO Layer, using a front-end function. Registration is required for two purposes:

1. The usermode task must have read only access to the *Virtual IO task* handle. This is needed to use the OS event notifications to notify the *Virtual IO task*. Therefore, one of the MPU regions of the task must be run-time configured to read-only access to *Virtual IO task* handler.

2. Usermode tasks are not authorized to use interrupt handlers, because interrupt handler code is executed in privileged mode. We used a queue system to communicate from interrupt handlers to

usermode tasks. Hence the registration routine creates a new queue and saves the queue handler in a structure. This will be used afterwards if the task will request access to one peripheral in interrupt mode.

The registration takes place through a system call that was previously mentioned in subsection 5.3.2, hidden by a front-end call. The system call is needed to configure an MPU region described in the former purpose. The registration procedure works as follows: (1) The usermode task invokes *the IO_Layer_init()* routine, which through (2) the IO_Layer_REGISTER system call (3) sets an MPU region of the caller task to access to *Virtual IO Task* descriptor in read-only mode. This is needed to send notifications. Then the framework creates and initializes a system queue (4) for using the DMA (the procedure is described in Back End Subset subsection). Before returning, if the procedure was successful, the task is added to the list of Virtual IO subscribed tasks.

Front End Subset

The Front End subset is intended to be called from the usermode tasks. These calls have the same signature of the original HAL library calls, beside the function name, which is extended with a prefix to make the programmer aware that is using the Virtual IO Layer and, obviously, to avoid a name space conflict. Thus for each HAL library function that we want to expose to the third party developer a function must be written. Each function declares a structure that contains:

1. The usermode task task handler.
2. A pointer to the relative back-end function to be called by the *Virtual IO Task*

3. A pointer for each original HAL Library function argument.

4. If the original HAL function returns a non-void value, a field to store it.

We refer to this structure with the name HAL Library Argument Embedding Structure (HAE Structure). Then HAE structure is instantiated in the Front End function, on the stack, and all structure's fields are assigned with their values. A notification is sent to the Virtual IO Layer Task with a pointer to this structure. At the end, optionally the HAL Library return value is returned if the function is non-void. A recap of the embedding of this function is shown in right top corner of Figure 5.2.

Back End Subset

The Back End (or call back functions) is the part of the library meant to be called by the *Virtual IO Task*. For each Front End function, there is one corresponding Back End one that takes as input a single argument, a void pointer. Its body contains a declaration of the HAE structure equal to the corresponding Front End function. The void pointer is then cast in this structure, arguments are then used to call the original HAL function. When the HAL Library call ends, the return argument is written in the structure, that still resides in the usermode stack. Finally the *Virtual IO Task* suspends its execution waiting for the next call and control returns to the usermode task.

This architecture has two advantages: (1) the ease of use, the programmer does not need to learn a new interface to use the HAL. (2) All Front End calls and Back End calls have the same format, so they can be written by a programmer or generated by an automatic tool, given the list of HAL functions that the *Virtual IO Library* will support.

To handle DMA asynchronous calls and to get notified when a DMA transfer is completed, we use the Queue returned when the usermode task subscribes the Virtual IO Layer. For security it is important that all the interrupt service routines (ISR) are implemented by the system. Moreover inside each service routine there is a Queue Send operation used to notify the task that wants to use the DMA that the routine is called. To correctly notify the corresponding usermode queue a reference table is used. This reference table is set by the back-end, when the usermode task invokes one of the DMA HAL Library functions.

Virtual IO Task

The *Virtual IO Task* is a privileged task that handles the communication from usermode tasks to peripherals. It starts when the Virtual IO layer is initialized, typically at system boot time. The communication is handled via *Direct Task Notification*. When started this task hangs in suspended state waiting for a call from one of the usermode registered tasks through the Front End.

The priority of this task is higher than all usermode tasks. Thus, when the notification is thrown from the Front End, the usermode task waits that the *Virtual IO task* ends its execution. Therefore, even if task notifications are asynchronous, the call to HAL Library is blocking because in FreeRTOS the preemption of the scheduler is priority based.

The body of this task, besides the *Task Notify Wait*, consists of an Access Control List (ACL), shown in Figure 5.2, that checks that the callee HAL Library function can be invoked by the caller. The pointer to HAE Structure is cast to a generic structure common for all HAE

Structures (we always know that the first two fields are fixed: the user-mode task task handler and the pointer to the call-back function), then the ACL permission check occurs. if the checking passed, the Back End function is invoked.

5.3.4 Dynamic Linking

The dynamic linking permits to execute new tasks without rebooting the system and enables the usage of systems resources from dynamic linked tasks. Thus, we implemented a privileged task in charge of dynamic linking other tasks named *Dynamic Linker task*. Run-time linked tasks must be cross compiled and have relocation and position independent compiler flags enabled. The *Dynamic Linker task* resolves at runtime unresolved dependencies to (1) system library functions (jump slots) and (2) global data declared in the system firmware. Once all dependencies are resolved a new FreeRTOS TCB is created and added to the ready task scheduler queue. The library in charge of dynamic linking usermode tasks is derived from the work of [134] and the dynamic linking consists of 3 steps:

1. **Allocation of Dynamic Linked Task**. The task sections are allocated in RAM.

2. **Relocation of *jump slots* and *global data***. Resolution of jump and data dependencies that points to the system firmware.

3. **FreeRTOS task creation and start**. Creation of the FreeRTOS task. The entry point of the task is set to a known and predefined function name.

To resolve the dependencies two sections of the system firmware ELF must be stored into Flash memory: *.symtab* and *.strtab*. The *Dynamic Linker Task* uses these sections to correctly relocate jump slots and global data to their real memory addresses. The dynamic linked task can be stored in Flash or RAM memory before being run-time linked.

5.4 Experimental Results

In this section we present results of Virtual IO Layer and Dyanamic Linking. All tests were conducted on an $STM32F411RE$ NUCLEO-64 Board [135]. This is a platform by ST Microelectronics, it embeds an ARM® 32-bit Cortex®-M4 CPU running up to 100 MHz with FPU and MPU. It features 512 KB of Flash memory and 128 KB of RAM memory. In our software setup we use the new driver for accessing hardware peripherals provided by ST called Hardware Abstraction Layer Driver (HAL Driver) [136].

5.4.1 Virtual IO Layer

We identified two main use cases, i.e. ways to access peripherals in a Microcontroller unit, that must be considered separately:

1. Atomic Action:

   In this case a HAL Driver routine is called each time we access a peripheral. In other words, the call does not involve data transfers after it, either if we access an IO address once, or if we access it in a loop. An example of this behavior is when we want to configure or read a GPIO PIN, or write something on the UART.

2. Continuous Action (or Tunneling Action):

   In this second case we consider all the peripherals that involve the use of DMA. For example when we want to set Analog to Digital converter and read it at regular intervals by the DMA.

Virtual IO Layer Timing

The time of accessing a peripheral using the Virtual IO Layer is reported in Table 5.2. The first row gives the cycles to get the task handle through a system call. The *MPU_xTaskGenericNotify()* is the direct task notification system call. The third row reports the cycles required to notify the *Virtual IO Task*. The last row gives the number of cycles to return control, after the HAL Driver call back to the User mode task. The cycles measurement has been done with the *DWT_CYCCNT* hardware cycle counter, available in Cortex-M4 MCUs.

Virtualization Step	**VIO** (Cycles)
getTaskHandle	97
MPU_xTaskGenericNotify	47
xTaskNotify + CS	490
Notify wait + CS back	293
TOTAL	**927**

Table 5.2: Timing overhead of accessing the IO using the Virtual IO Layer in Cycles

It is worth mentioning that with this paradigm, continuous mode operations pay the overhead just once, when the setup of the peripheral or IO is performed. Thus when the DMA is working the only overhead is the queue used to synchronize the ISR with the user mode task.

The cycles overhead to check if the function that the user mode task wants to use is permitted by the ACL grows linearly with the number of

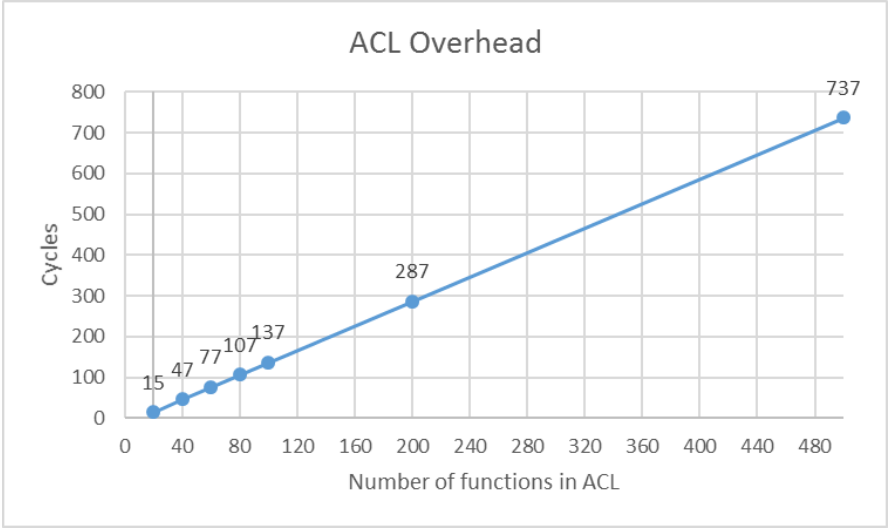


Figure 5.3: Overhead of the control in the ACL.

checks that occurs. In Figure 5.3, the overhead is reported. As expected the number of cycles are proportional to the number of function addresses to verify.

Virtual IO Layer Memory Footprint

The overhead in terms of memory footprint is described in Table 5.3. We show the code size of the library and of the Virtual IO Task separately, in case the compiler is invoked with the flag for performance (-O3) or space (-OS) optimization. The Size of the Virtual IO Library is measured with an average size of 50 functions (front end + back end). As we can notice from the results, the memory footprint is minimal. Moreover we notice that optimizing for space and performance gives the same overhead, it is due to the fact that employing space optimization means introducing -O2 compiler optimization as well. The code is not computationally intensive thus -O2 and -O3 produce the same timing overhead for accessing the peripheral.

Opt.	VIO Task	VIO Library	Overhead
-O3	592 B	2876 B	927 Cycles
-OS	464 B	2314 B	927 Cycles

Table 5.3: Virtualization Layer code size and access overhead with relation to space and performance optimization (Opt.)

5.4.2 Dynamic Linking

The cycles needed to link dynamically a task are dependent from the number of relocations. As Section 5.3.4 describes, two possible kind of relocations are supported, global data and function call relocation, the cost of both is equal and dependent to the cycles needed to find the entry in the system firmware elf.

Single Relocation		
Dynamic Link Step	**KCycles**	**ms @ 100MhZ**
Relocation	181	1.81
Allocation and Start	19	0.19
TOTAL	**200**	**2.00**

Table 5.4: Dynamic Linker Cycles

In Table 5.4 we show the dynamic linking execution cycles, we averaged the cycles of 10 global data and 10 function relocations. The *allocation* and *task creation and start* described in Section 5.3.4 are grouped in one entry, while the other entry shows the *relocation*. It is worth to say that the relocation cycles are required for each additional variable or function to relocate, while the task allocation and start cycles are payed just once per task link. This means that if we want to relocate 100 variables and functions we pay an average of 18.1 $MCycles$ that at 100 MhZ are 181 ms. Dynamic linked tasks usually have a number much lower than 100 relocations for a single task since they use a limited number of calls to system firmware functions or global data.

Opt.	Code Size	Reloc.	Alloc. + Start
-O3	9904 B	181 KCycles	19 KCycles
-OS	6592 B	190 KCycles	21 KCycles

Table 5.5: Dynamic Linker code size and performance with relation to the compiler optimization (Opt.)

In Table 5.5 we show the memory footprint of the dynamic linking system in code size. As in Section 5.4.1 we measured the overhead using compiler option for space (-OS) and performance optimization (-O3). The memory footprint of the dynamic linker is higher than the Virtual IO Layer but still limited. Moreover we show the relocation cycles of both compiler optimization flags. With space optimization we save roughly 30% in space, paying a very limited amount of overhead cycles.

Finally we implemented a task that samples with the ADC an accelerometer and, after a FIR filter stage, sends the results through the WIFI to a remote cloud. We tested it in two versions: one statically written in the system firmware, in a standard FreeRTOS with neither virtualization nor dynamic linking. The second version uses our enhanced FreeRTOS with Virtual IO and dynamic linking to link a new filter runtime. After an initialization stage, different for the two implementations, the main loop of the task (1) waits a notification from the DMA ISR, (2) collects and elaborates the samples, and (3) sends them to a third task that collects results to forward through the WIFI. *Step 2* is exactly the same for both implementations and it is responsible for the majority of the execution time, the filter elaborates 512 samples each 10 ms and the filtering takes 523 $KCycles$ (5,23 ms at 100 MhZ). *Step 1* costs 104 $Cycles$ (1.04 μs at 100 MhZ) and *Step 3* costs 512 $Cycles$ (5.12 μs at 100 MhZ) for the static task, while for the task that uses the infrastructure

presented in this chapter they have an overhead of 101 *Cycles* each since they are implemented within system calls. The percentage increase for both *Step 1* and *Step 3* is 32%, while considering all 3 Steps the overhead of using our runtime system is really small, only 0.03% compared to the static one, with all advantages discussed in previous Sections. A summary is presented in Table 5.6.

Main Loop	**Native**	**Our Solution**	**Overhead**
Wait data transfer	104 Cycles	205 Cycles	97%
FIR Filtering	523 KCycles	523 KCycles	0%
Wait data transfer	512 Cycles	613 Cycles	97%
	Total overhead		**0.03%**

Table 5.6: Native vs our solution overhead

As a concluding note, it is important to notice the fact that the runtime execution of tasks, when not interacting with the IOs or using system calls, is exactly the same as native FreeRTOS tasks, with no performance overhead for memory protection; as the MPU is completely transparent from the performance viewpoint. This is very similar to what happens in virtual machine execution for high-end cores, and in sharp contrast with interpreted virtual machines or even JIT-based systems.

5.5 Conclusions

In this chapter we presented a virtualization layer for low-cost microcontrollers which creates a separation between kernel mode and user mode and protects the hardware resources from misuses when concurrent tasks or function are written by different developers.

Moreover we demonstrated the effectiveness of a mechanism capable to execute new runtime code, without the need of system reboot. We have

focused on small size of the framework and on lower overhead, because targeted for low-cost and limited computing capabilities microcontrollers such as the ones designed for IoT and WSN. Experimental results demonstrate that the overhead is limited and time delay is negligible considering the typical application scenarios.

Conclusions

Computer Vision on Embedded Systems is a challenging research topic. In this thesis we provided solutions to critical missing links between these two worlds, studying novel algorithms and providing along with the accuracy measurements a performance analysis for a target embedded system. Three challenging scenarios were presented, dealing with images and videos in high-end mobile devices and low-end MCUs and finally proposing a framework that address flexibility and security on MPU enabled Microcontrollers.

The first contribution proposed a **people counting algorithm in mostly static context** integrated in IoT WSN with heterogeneous nodes. It elaborates images, instead of videos, to reduce the overhead of computation and network traffic. Our results prove that a good trade-off between required computational power and accuracy can be reached for the proposed target to combine occupancy and micro-climatic information.

After dealing with stationary computer vision, we presented a novel **ego-centric gesture recognition algorithm** for monocular wearable cameras. It is able to recognize with a good precision static and dynamic gestures and can achieve high accuracy results even when trained with a few positive samples. Results are shown on publicly available datasets

to compare our solution to existing solutions in literature.

Therefore the **gesture recognition has been fully integrated in wearable vision system** to enhance visitors' museum experiences. We proposed a networked architecture capable of recognizing users' gestures and artworks. We fully integrated the gesture recognition on an ARM big.LITTLE heterogeneous platform for embedded devices and we implemented a tradeoff between accuracy and performance for a virtual (Interactive Museum dataset) and a real (Maramotti Dataset) museum environments. Results show the feasibility of our solution on a real-world scenario.

The fourth contribution proposed a **context change detector for a low-resolution ultra-low power imager**. We proposed a solution based on a Siamese Convolutional Neural Network, that is able to work with grayscale low-resolution snapshots. We finally ran extensive experiments showing the robustness and efficacy of the proposed method with respect to related approaches.

Finally we dealt with low-end MCUs security and flexibility, presenting a **lightweight virtualization framework** able to protect peripherals to misuses and solution the dynamic loading of new user code. We have focused on small size of the framework because targeted for low-cost and limited computing capabilities microcontrollers. Experimental results demonstrate that the overhead is limited and time delay is negligible considering the typical application scenario.

The contributions proposed in this thesis want to suggest a growing constraints paradigm. We first presented a stationary high-end scenario, then a wearable high-end and low-end one. In particular the last two contributions, presented in Chapter 4 and Chapter 5, with the exploitation

and study of ultra-low-power low-resolution imagers and a flexible and secure deployment framework want to take a step toward a transition from computer vision to always-on *sensor vision*. The low-power consumption given by the exploitation of low-end devices, combined with the use of energy harvesters and ultra-low-power low-resolution imagers permits to save computational power, energy and to deal with privacy issues that is one important concerns in wearable computer vision. Moreover the flexibility and security of third-party application deployment can permit a wider and easier diffusion of these devices. Obviously this does not come for free, the study of more efficient algorithms and architectures and the exploitation of low-resolution images rises the difficulty of these scenarios. A possible future perspective is to employ such a solution as one of the available sensor peripherals in an embedded system with high energy efficiency that can be kept always-on to collect and elaborate information or be used as a trigger for high resolution and more power-hungry subsystems.

List of Publications

1. **Paci, Francesco**, Davide Brunelli, Luca Benini. *"0, 1, 2, many - A Classroom Occupancy Monitoring System for Smart Public Buildings."* Proceedings of the Conference on Design and Architectures for Signal and Image Processing (DASIP), Madrid, Spain, IEEE, 2014

2. Baraldi, Lorenzo, **Francesco Paci**, Giuseppe Serra, Luca Benini, Rita Cucchiara. *"Gesture Recognition in Ego-Centric Videos using Dense Trajectories and Hand Segmentation."* Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) Workshops, Columbus, Ohio, IEEE, 2014.

3. Baraldi, Lorenzo, **Francesco Paci**, Giuseppe Serra, Luca Benini, Rita Cucchiara. *"Gesture Recognition Using Wearable Vision Sensors to Enhance Visitors' Museum Experiences."* IEEE Sensors Journal, May 2015.

4. **Paci, Francesco**, Lorenzo Baraldi, Giuseppe Serra, Rita Cucchiara, Luca Benini. *"Context Change Detection for an Ultra-Low Power Low-Resolution Ego-Vision Imager."* Proceedings of the 14th European Conference on Computer Vision Workshops, Amsterdam, The Netherlands, 2016

5. **Paci, Francesco**, Davide Brunelli, Luca Benini. *"Lightweight IO Virtualization On MPU Enabled Microcontrollers."* Proceedings of the 4th Embedded Operating Systems Workshop, Pittsburgh, USA, 2016

Acknowledgments

Foremost, I would like to thank Dr. Elisa Ricci and Prof. Geoff Merrett that reviewed this Thesis for their precious advices.

I would like to express my sincere gratitude to my PhD Advisor, Prof. Luca Benini, for the possibility he gave me to work in his group, for his high curiosity and enthusiasm on new projects and on supporting and discussing new ideas. Moreover to patiently teach me scientific rigor and for his discuss-once policy, i.e. once something is agreed we go straight to the target. I would like to thank whole Bologna and Zürich group, a special mention to Dr. Davide Brunelli, for the work done together over these years.

I would also like to thank my co-advisors Prof. Rita Cucchiara and Prof. Michela Milano. A specials thanks goes to all Prof. Rita Cucchiara's Group, ImageLab, at University of Modena e Reggio Emilia, in Particular to Dr. Giuseppe Serra, which patiently followed me during these years and to Lorenzo Baraldi.

Bibliography

[1] R. Szeliski, *Computer Vision*. Springer-Verlag London, 2011.

[2] R. F. Lyon, "The optical mouse: Early biomimetic embedded vision," in *Advances in Embedded Computer Vision*, B. Kisačanin and M. Gelautz, Eds. Springer International Publishing, 2014, ch. 1, pp. 3–22.

[3] L. H. Matthies, M. W. Maimone, A. E. Johnson, Y. Cheng, R. G. Willson, C. Villalpando, S. B. Goldberg, A. Huertas, A. N. Stein, and A. Angelova, "Computer vision on mars," *International Journal of Computer Vision*, vol. 75, pp. 67–92, 2007.

[4] M. Camplani, T. Mantecón, and L. Salgado, "Depth-color fusion strategy for 3-d scene modeling with kinect," *IEEE Trans. Cybernetics*, vol. 43, pp. 1560–1571, 2013.

[5] V. Kumar and E. Todorov, "Mujoco haptix: A virtual reality system for hand manipulation," in *Humanoids*, 2015.

[6] B. Besbes, A. Rogozan, A.-M. Rus, A. Bensrhair, and A. Broggi, "Pedestrian detection in far-infrared daytime images using a hierarchical codebook of surf," in *Sensors*, 2015.

[7] R. Tapia-Espinoza and M. Torres-Torriti, "Robust lane sensing and departure warning under shadows and occlusions," in *Sensors*, 2013.

[8] S. Eum and H. G. Jung, "Enhancing light blob detection for intelligent headlight control using lane detection," *IEEE Trans. Intelligent Transportation Systems*, vol. 14, pp. 1003–1011, 2013.

[9] Y. Zhang, X. Xu, H. Lu, and Y. Dai, "Two-stage obstacle detection based on stereo vision in unstructured environment," in *Intelligent Human-Machine Systems and Cybernetics (IHMSC), 2014 Sixth International Conference on*, vol. 1. IEEE, 2014, pp. 168–172.

[10] Y.-L. Chen, B.-F. Wu, H.-Y. Huang, and C.-J. Fan, "A real-time vision system for nighttime vehicle detection and traffic surveillance," *IEEE Trans. Industrial Electronics*, vol. 58, pp. 2030–2044, 2011.

[11] A. Saeed, A. Al-Hamadi, A. Ghoneim, and V. M. N. Passaro, "Head pose estimation on top of haar-like face detection: A study using the kinect sensor," in *Sensors*, 2015.

[12] W. J. MacLean, "An evaluation of the suitability of fpgas for embedded vision systems," in *CVPR Workshops*, 2005.

[13] A. Darabiha, J. Rose, and W. J. MacLean, "Video-rate stereo depth measurement on programmable hardware," in *CVPR*, 2003.

[14] C. Ttofis, C. Kyrkou, and T. Theocharides, "A hardware-efficient architecture for accurate real-time disparity map estimation," *ACM Trans. Embedded Comput. Syst.*, vol. 14, pp. 36:1–36:26, 2015.

[15] D. Moloney, "1 tops/w software programmable media processor," in *2011 IEEE Hot Chips 23 Symposium (HCS)*, Aug 2011, pp. 1–24.

[16] J. S. Z. Lin and T. Flanagan, "Empowering automotive vision with tis vision accelerationpac," in *TI White Paper*, 2013.

[17] D. Honegger, L. Meier, P. Tanskanen, and M. Pollefeys, "An open source and open hardware embedded metric optical flow cmos camera for indoor and outdoor applications," in *2013 IEEE International Conference on Robotics and Automation*, May 2013, pp. 1736–1741.

[18] Open Machine Vision. Accessed: 2016-12-16. [Online]. Available: https://openmv.io/

[19] C. Gurrin, A. F. Smeaton, and A. R. Doherty, "Lifelogging: Personal big data," *Found. Trends Inf. Retr.*, vol. 8, no. 1, pp. 1–125, Jun. 2014. [Online]. Available: http://dx.doi.org/10.1561/1500000033

[20] M. Rusci, D. Rossi, M. Lecca, M. Gottardi, E. Farella, and L. Benini, "An event-driven ultra-low-power smart visual sensor," *IEEE Sensors Journal*, vol. 16, no. 13, pp. 5344–5353, July 2016.

[21] A. Berkovich, M. Lecca, L. Gasparini, P. A. Abshire, and M. Gottardi, "A 30 μw 30 fps 110$\times$ 110 pixels vision sensor embedding local binary patterns," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 9, pp. 2138–2148, 2015.

[22] L. Gasparini, R. Manduchi, M. Gottardi, and D. Petri, "An ultralow-power wireless camera node: Development and performance analysis," *IEEE Trans. Instrumentation and Measurement*, vol. 60, pp. 3824–3832, 2011.

[23] A. Wardhani and B. Pham, "Programming optimisation for embedded vision," 2007.

[24] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, pp. 2274–2282, 2012.

[25] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *ECCV*, 2010.

[26] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *ECCV*, 2006.

[27] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "Orb: An efficient alternative to sift or surf," in *ICCV*, 2011.

[28] P. A. Viola and M. J. Jones, "Robust real-time face detection," *International Journal of Computer Vision*, vol. 57, pp. 137–154, 2001.

[29] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *ECCV*, 2006.

[30] K. Matsuo, K. Yamada, S. Ueno, and S. Naito, "An attention-based activity recognition for egocentric video," in *CVPR Workshops*, 2014.

[31] Y. Yan, E. Ricci, G. Liu, and N. Sebe, "Egocentric daily activity recognition via multitask clustering," *IEEE Transactions on Image Processing*, vol. 24, no. 10, pp. 2984–2995, Oct 2015.

[32] S. Singh, C. Arora, and C. V. Jawahar, "Trajectory aligned features for first person action recognition," *CoRR*, vol. abs/1604.02115, 2017.

[33] K. Yamada, Y. Sugano, T. Okabe, Y. Sato, A. Sugimoto, and K. Hiraki, "Attention prediction in egocentric video using motion and visual saliency," in *PSIVT*, 2011.

[34] Y.-C. Su and K. Grauman, "Detecting engagement in egocentric video," in *Proc. of the European Conference on Computer Vision*, 2016.

[35] V. Buso, J. Benois-Pineau, and J.-P. Domenger, "Geometrical cues in visual saliency models for active object recognition in egocentric videos," in *Multimedia Tools and Applications*, 2014.

[36] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[37] M. Magno, D. Boyle, D. Brunelli, B. O'Flynn, E. Popovici, and L. Benini, "Extended wireless monitoring through intelligent hybrid energy supply," *Industrial Electronics, IEEE Transactions on*, vol. 61, no. 4, pp. 1871–1881, April 2014.

[38] "IBM - smarter planet - smarter buildings - solutions - energy optimization - united states," Dec. 2012. [Online]. Available: https://www.ibm.com/smarterplanet/us/en/green_buildings/nextsteps/solution/U481210X53127D39.html

[39] "Building automation systems - building technologies - siemens." [Online]. Available: http://www.buildingtechnologies.siemens.com/bt/global/en/buildingautomation-hvac/building-automation/pages/building-automation-system.aspx

[40] D. Porcarelli, D. Balsamo, D. Brunelli, and G. Paci, "Perpetual and low-cost power meter for monitoring residential and industrial appliances," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, March 2013, pp. 1155–1160.

[41] F. Manshadi, B. Awerbuch, R. Gemulla, R. Khandekar, J. Mestre, and M. Sozio, "A distributed algorithm for large-scale generalized matching," *Proceedings of the VLDB Endowment*, 2013.

[42] J. W. Branch, C. Giannella, B. Szymanski, R. Wolff, and H. Kargupta, "In-network outlier detection in wireless sensor networks," *Knowledge and information systems*, vol. 34, no. 1, pp. 23–54, 2013.

[43] C. Caione, D. Brunelli, and L. Benini, "Compressive sensing optimization for signal ensembles in wsns," *Industrial Informatics, IEEE Transactions on*, vol. 10, no. 1, pp. 382–392, Feb 2014.

[44] M. Andriluka, S. Roth, and B. Schiele, "People-tracking-by-detection and people-detection-by-tracking," in *IEEE Conference on Computer Vision and Pattern Recognition, 2008. CVPR 2008*, 2008, pp. 1–8.

[45] M. Breitenstein, F. Reichlin, B. Leibe, E. Koller-Meier, and L. Van Gool, "Online multiperson tracking-by-detection from a single, uncalibrated camera," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 9, pp. 1820–1833, 2011.

[46] T. Li, H. Chang, M. Wang, B. Ni, R. Hong, and S. Yan, "Crowded scene analysis: A survey," *CoRR*, vol. abs/1502.01812, 2015.

[47] B. Zhan, D. N. Monekosso, P. Remagnino, S. A. Velastin, and L.-Q. Xu, "Crowd analysis: a survey," *Machine Vision and Applications*, vol. 19, no. 5, pp. 345–357, 2008. [Online]. Available: http://dx.doi.org/10.1007/s00138-008-0132-4

[48] Y.-L. Hou and G. Pang, "People counting and human detection in a challenging situation," *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, vol. 41, no. 1, pp. 24–33, 2011.

[49] J. Luo, J. Wang, H. Xu, and H. Lu, "Real-time people counting for indoor scenes," *Signal Processing*, vol. 124, pp. 27–35, 2016.

[50] Q. Ye, "A robust method for counting people in complex indoor spaces," in *2010 2nd International Conference on Education Technology and Computer (ICETC)*, vol. 2, 2010, pp. V2–450–V2–454.

[51] Q. Ye and Z. Yang, "A method of automatic people counting used in air-conditioning energy-saving," in *2010 2nd International Conference on Computer Engineering and Technology (ICCET)*, vol. 6, 2010, pp. V6–703–V6–708.

[52] K.-Z. Lee, L.-W. Tsai, and P.-C. Hung, "Fast people counting using sampled motion statistics," in *2012 Eighth International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP)*, 2012, pp. 162–165.

[53] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Video-Based Surveillance Systems*, P. Remagnino, G. A. Jones, N. Paragios, and C. S. Regazzoni, Eds. Springer US, jan 2002, pp. 135–144.

[54] T. Teixeira and A. Savvides, "Lightweight people counting and localizing for easily deployable indoors WSNs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 2, no. 4, pp. 493–502, 2008.

[55] "Data sheet: W24th wireless sensor node, wispes srl." [Online]. Available: http://www.wispes.com

[56] "Data sheet: Jn5148-001, ieee 802.15.4 wireless microcontroller, jennic, u.k." [Online]. Available: http://www.jennic.com/files/support_files/JN-DS-JN5148-1v8.pdf

[57] S. Sudevalayam and P. Kulkarni, "Energy harvesting sensor nodes: Survey and implications," *Communications Surveys & Tutorials, IEEE*, vol. 13, no. 3, pp. 443–461, 2011.

[58] T. Kanade and M. Hebert, "First-person vision," *Proceedings of the IEEE*, vol. 100, no. 8, pp. 2442–2453, Aug 2012.

[59] H. Pirsiavash and D. Ramanan, "Detecting activities of daily living in first-person camera views," in *Proc. of CVPR*, 2012.

[60] S. Sundaram and W. W. M. Cuevas, "High level activity recognition using low resolution wearable vision," in *Proc. of CVPR*, 2009.

[61] A. Fathi, X. Ren, and J. M. Rehg, "Learning to recognize objects in egocentric activities," in *Proc. of CVPR*, 2011.

[62] A. Fathi and J. M. Rehg, "Modeling actions through state changes," in *Proc. of CVPR*, 2013.

[63] R. Khan, A. Hanbury, and J. Stoettinger, "Skin detection: A random forest approach," in *Proc. of ICIP*, 2010.

[64] C. Li and K. M. Kitani, "Pixel-level hand detection in ego-centric videos," in *Proc. of CVPR*, 2013.

[65] A. Sanin, C. Sanderson, M. T. Harandi, and B. C. Lovell, "Spatio-temporal covariance descriptors for action and gesture recognition," in *Proc. of Workshop on Applications of Computer Vision*, 2013.

[66] Y. M. Lui, J. R. Beveridge, and M. Kirby, "Action classification on product manifolds," in *Proc. of CVPR*, 2010.

[67] Y. M. Lui and J. R. Beveridge, "Tangent bundle for human action recognition," in *In proc. of Automatic Face & Gesture Recognition and Workshops*, 2011.

[68] T.-K. Kim and R. Cipolla, "Canonical correlation analysis of video volume tensors for action categorization and detection," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 31, no. 8, pp. 1415–1428, 2009.

[69] P. Mistry and P. Maes, "Sixthsense: A wearable gestural interface," in *ACM SIGGRAPH ASIA 2009 Sketches*. ACM, 2009, pp. 11:1–11:1.

[70] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu, "Action Recognition by Dense Trajectories," in *Proc. of CVPR*, 2011.

[71] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, vol. 1. IEEE, 2005, pp. 886–893.

[72] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *Computer Vision–ECCV 2006*. Springer, 2006, pp. 428–441.

[73] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[74] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Proc. of ECCV*, 2006.

[75] G. Farnebäck, "Two-frame motion estimation based on polynomial expansion," in *Image Analysis*. Springer, 2003, pp. 363–370.

[76] C. Zach, T. Pock, and H. Bischof, "A duality based approach for realtime tv-l 1 optical flow," in *Pattern Recognition*. Springer, 2007, pp. 214–223.

[77] M. Tao, J. Bai, P. Kohli, and S. Paris, "Simpleflow: A non-iterative, sublinear optical flow algorithm," in *Computer Graphics Forum*, vol. 31, no. 2pt1. Wiley Online Library, 2012, pp. 345–353.

[78] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," in *Proc. of ECCV*, 2010.

[79] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Susstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 34, no. 11, pp. 2274–2282, 2012.

[80] C. Rother, V. Kolmogorov, and A. Blake, "Grabcut: Interactive foreground extraction using iterated graph cuts," in *ACM Transactions on Graphics (TOG)*, vol. 23, no. 3. ACM, 2004, pp. 309–314.

[81] T.-K. Kim, K.-Y. K. Wong, and R. Cipolla, "Tensor canonical correlation analysis for action classification," in *Proc. of CVPR*, 2007.

[82] E. Hayman and J.-O. Eklundh, "Statistical background subtraction for a mobile observer," in *Proc. of ICCV*, 2003.

[83] M. J. Jones and J. M. Rehg, "Statistical color models with application to skin detection," *International Journal of Computer Vision*, vol. 46, no. 1, pp. 81–96, 2002. [Online]. Available: http://dx.doi.org/10.1023/A:1013200319198

[84] "Odroid-XU developement board by Hardkernel," http://www.hardkernel.com.

[85] "Samsung Exynos5 5410 ARM SoC," http://www.samsung.com/global/business/semiconductor/minisite/Exynos/products5octa_5410.html.

[86] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *in Proc ICCV*, 2003.

[87] A. D. Bagdanov, L. Ballan, M. Bertini, and A. Del Bimbo, "Trademark matching and retrieval in sports video databases," in *Proc.*

of ACM International Workshop on Multimedia Information Retrieval (MIR), 2007.

[88] P. Kelly, A. Doherty, E. Berry, S. Hodges, A. M. Batterham, and C. Foster, "Can we use digital life-log images to investigate active and sedentary travel behaviour? results from a pilot study," *International Journal of Behavioral Nutrition and Physical Activity*, vol. 8, no. 1, p. 1, 2011.

[89] H. Zhang, L. Li, W. Jia, J. D. Fernstrom, R. J. Sclabassi, and M. Sun, "Recognizing physical activity from ego-motion of a camera," in *Proc. of the IEEE Conference Engineering in Medicine and Biology*, 2010.

[90] S. Hodges, L. Williams, E. Berry, S. Izadi, J. Srinivasan, A. Butler, G. Smyth, N. Kapur, and K. Wood, "Sensecam: A retrospective memory aid," in *Proc. of the International Conference on Ubiquitous Computing*. Springer, 2006.

[91] A. M. Khan, A. Tufail, A. M. Khattak, and T. H. Laine, "Activity recognition on smartphones via sensor-fusion and kda-based svms," *International Journal of Distributed Sensor Networks*, vol. 2014, 2014.

[92] S. Patel, H. Park, P. Bonato, L. Chan, and M. Rodgers, "A review of wearable sensors and systems with application in rehabilitation," *Journal of neuroengineering and rehabilitation*, vol. 9, no. 1, p. 1, 2012.

[93] A. Mannini and A. M. Sabatini, "Machine learning methods for classifying human physical activity from on-body accelerometers," *Sensors*, vol. 10, no. 2, pp. 1154–1175, 2010.

[94] J. R. Kwapisz, G. M. Weiss, and S. A. Moore, "Activity recognition using cell phone accelerometers," *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74–82, 2011.

[95] D. Guan, W. Yuan, Y.-K. Lee, A. Gavrilov, and S. Lee, "Activity recognition based on semi-supervised learning," in *Proc. of the International Conference on Embedded and Real-Time Computing Systems and Applications*, 2007.

[96] K. K. Singh, K. Fatahalian, and A. A. Efros, "Krishnacam: Using a longitudinal, single-person, egocentric dataset for scene understanding tasks," in *Proc. of the IEEE Winter Conference on Applications of Computer Vision*, 2016.

[97] S. Alletto, G. Serra, S. Calderara, and R. Cucchiara, "Understanding social relationships in egocentric vision," *Pattern Recognition*, vol. 48, no. 12, pp. 4082–4096, 2015.

[98] Y. Li, A. Fathi, and J. M. Rehg, "Learning to predict gaze in egocentric video," in *Proc. of the International Conference on Computer Vision*, 2013.

[99] Y. Poleg, C. Arora, and S. Peleg, "Temporal segmentation of egocentric videos," in *Proc. of the Conference on Computer Vision and Pattern Recognition*, 2014.

[100] D. Castro, S. Hickson, V. Bettadapura, E. Thomaz, G. Abowd, H. Christensen, and I. Essa, "Predicting daily activities from egocentric images using deep learning," in *Proc. of the ACM International symposium on Wearable Computers*, 2015.

[101] M. S. Ryoo, B. Rothrock, and L. Matthies, "Pooled motion features for first-person videos," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 896–904.

[102] A. L. Wood, G. V. Merrett, S. R. Gunn, B. M. Al-Hashimi, N. R. Shadbolt, and W. Hall, "Adaptive sampling in context-aware systems: A machine learning approach," in *IET Conference on Wireless Sensor Systems (WSS 2012)*, June 2012, pp. 1–5.

[103] Z. Lu and K. Grauman, "Story-driven summarization for egocentric video," in *Proc of the Conference on Computer Vision and Pattern Recognition*, 2013.

[104] M. Dimiccoli, M. Bolaños, E. Talavera, M. Aghaei, S. G. Nikolov, and P. Radeva, "Sr-clustering: Semantic regularized clustering for egocentric photo streams segmentation," *arXiv preprint arXiv:1512.07143*, 2015.

[105] A. Bifet and R. Gavalda, "Learning from time-changing data with adaptive windowing." in *Proc. of International Conference on Data Mining*, 2007.

[106] M. Magno, D. Brunelli, L. Sigrist, R. Andri, L. Cavigelli, A. Gomez, and L. Benini, "Infinitime: Multi-sensor wearable bracelet with human body harvesting," *Sustainable Computing:*

Informatics and Systems, pp. –, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2210537916300816

[107] MSP430, "by texa instruments," http://www.ti.com/ww/it/msp430.html, [Online; accessed 15-July-2016].

[108] Centeye, http://www.centeye.com/, [Online; accessed 15-July-2016].

[109] L. Spadaro, M. Magno, and L. Benini, "Kinetisee: A perpetual wearable camera acquisition system with a kinetic harvester: Poster abstract," in *Proceedings of the 15th International Conference on Information Processing in Sensor Networks*, ser. IPSN '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 68:1–68:2. [Online]. Available: http://dl.acm.org/citation.cfm?id=2959355.2959423

[110] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[111] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[112] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition." in *Proc. of the International Conference on Machine Learning*, 2014.

[113] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015.

[114] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2, 2006, pp. 1735–1742.

[115] L. Baraldi, C. Grana, and R. Cucchiara, "A deep siamese network for scene detection in broadcast videos," in *Proceedings of the 23rd ACM international conference on Multimedia*. ACM, 2015, pp. 1199–1202.

[116] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005.

[117] Narrative, http://getnarrative.com/, [Online; accessed 15-July-2016].

[118] F. Conti and L. Benini, "A ultra-low-energy convolution engine for fast brain-inspired vision in multicore clusters," in *DATE 2015*, 2015.

[119] P. Gysel, M. Motamedi, and S. Ghiasi, "Hardware-oriented approximation of convolutional neural networks," *CoRR*, vol. abs/1604.03168, 2016. [Online]. Available: http://arxiv.org/abs/1604.03168

[120] "Arm virtualization extension." [Online]. Available: https://www.arm.com/products/processors/technologies/virtualization-extensions.php

[121] *ARM Security Technology - Building a Secure System using TrustZone Technology*, whitepaper, April 2009.

[122] T. Alves and D. Felton, *Trustzone: Integrated hardware and software security-enabling trusted computing in embedded systems*, white paper, arm, july 2004.

[123] "Micropython website." [Online]. Available: http://micropython.org/

[124] "Pymite." [Online]. Available: https://wiki.python.org/moin/PyMite

[125] "Oracle java me embedded." [Online]. Available: http://www.oracle.com/us/technologies/java/embedded/micro-edition/overview/index.html

[126] N. B. *et al.* , "Darjeeling, a feature-rich vm for the resource poor," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, ser. SenSys '09. New York, NY, USA: ACM, 2009, pp. 169–182. [Online]. Available: http://doi.acm.org/10.1145/1644038.1644056

[127] "Espruino javascript interpreter." [Online]. Available: http://www.espruino.com/

[128] "Embedded power driven by lua." [Online]. Available: http://www.eluaproject.net/

[129] A. B. *et al.* , "Virtualsense: A java-based open platform for ultra-low-power wireless sensor nodes," *International Journal of Distributed Sensor Networks*, vol. 2012, 2012.

[130] "Contiki: The open source os for the internet of things," http://www.contiki-os.org/.

[131] S. B. *et al.* , "Mantis os: An embedded multithreaded operating system for wireless micro sensor platforms," *Mob. Netw. Appl.*, vol. 10, no. 4, pp. 563–579, Aug. 2005. [Online]. Available: http://dl.acm.org/citation.cfm?id=1160162.1160178

[132] M. P. A. *et al.* , "System design for a synergistic, low power mote/ble embedded platform," in *Proceedings of the 15th International Conference on Information Processing in Sensor Networks*, ser. IPSN '16. Piscataway, NJ, USA: IEEE Press, 2016, pp. 17:1–17:12. [Online]. Available: http://dl.acm.org/citation.cfm?id=2959355.2959372

[133] "Freertos website." [Online]. Available: http://www.freertos.org/

[134] S. Holmbacka, W. Lund, S. Lafond, and J. Lilius, "Lightweight framework for runtime updating of c-based software in embedded systems," in *Presented as part of the 5th Workshop on Hot Topics in Software Upgrades*. San Jose, CA: USENIX, 2013. [Online]. Available: https://www.usenix.org/conference/hotswup13/workshop-program/presentation/Holmbacka

[135] "St microelectronics nucleo boards." [Online]. Available: http://www.st.com/content/st_com/en/products/evaluation-tools/product-evaluation-tools/mcu-eval-tools/stm32-mcu-eval-tools/stm32-mcu-nucleo.html?querycriteria=productId=LN1847

[136] "St microelectronics hardware abstraction layer driver." [Online]. Available: http://www.st.com/content/ccc/resource/technical/document/user_manual/2f/71/ba/b8/

75/54/47/cf/DM00105879.pdf/files/DM00105879.pdf/jcr:
content/translations/en.DM00105879.pdf