TÉCNICO LISBOA

FPGA-Accelerated ORB System for Real-time 3D Mapping Applications

André Santiago Mestre e Costa

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors: Doutor Nuno Filipe Simões Santos Moraes da Silva Neves
Doutor Nuno Filipe Valentim Roma

Examination Committee

Chairperson: Doutor António Manuel Raminhos Cordeiro Grilo
Supervisor: Doutor Nuno Filipe Simões Santos Moraes da Silva Neves
Member of the Committee: Doutor José João Henriques Teixeira de Sousa

December 2024

Declaração

Declaro que o presente documento é um trabalho original da minha autoria e que cumpre todos os requisitos do Código de Conduta e Boas Práticas da Universidade de Lisboa.

Declaration

I declare that this document is an original work of my own authorship and that it fulfils all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

While this thesis was written by me, it is the result of the efforts of many people. I have to thank my parents who gifted me the opportunity to pursue my dream to become an engineer, and the support to pursue whichever projects I found interesting throughout my student life. It is because of this freedom that I got to meet amazing people and develop myself both as an engineer and as an individual. The contribution of the professors I had at Técnico is also undeniably significant to the success of the work presented here. I extend my gratitude to the often demanding yet always passionate individuals who uphold the reputation of this institution.

Last but certainly not least, I would like to thank Nuno Neves, Nuno Roma, and Pedro Tomás for the opportunity to learn from them. Their constant availability, insightful guidance, freedom to make my own design decisions, and ongoing motivation to push the developments further were everything I could have hoped for.

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) under projects 3D-CAVE (DOI: 10.54499/2022.04020.PTDC) and UNIFY (DOI: 10.54499/2022.06780.PTDC). An acknowledgment is also in order for the contributions from Sociedade Portuguesa de Espeleologia (SPE) and its cave diver team for collecting the datasets used in this work, providing access to challenging underwater cave conditions.

Abstract

This thesis introduces a novel ORB (Oriented FAST and Rotated BRIEF) accelerator architecture tailored for low-power embedded Systems on Chip (SoCs), catering to the computational and energy efficiency needs of real-time feature extraction in embedded computer vision tasks. It deploys a configurable datapath, effectively balancing accuracy and computational complexity and offering application-specific trade-offs. The proposed architecture was also implemented on an embedded FPGA SoC, integrating the ORB accelerator on the reconfigurable fabric and deploying a Robot Operating System (ROS) node running on the CPU, forming a complete feature extraction device. The devised integration with ROS facilitates easy deployment into modular computer vision applications.

An experimental evaluation of the proposed ORB accelerator was conducted using images from the standard TUM-VI dataset to assess its accuracy, robustness to rotation, and energy efficiency. The obtained results demonstrate the capabilities of the proposed feature extraction architecture, meeting the performance of the state-of-the-art solutions while still attaining between 6.7x and 16.2x of energy efficiency gains.

Finally, the conceived device was also validated by using a real underwater cave dataset in a real-time demonstration of feature overlay on a video being streamed to the feature extraction device, further establishing this work as a potential solution for practical applications.

Keywords

Hardware Accelerator; Embedded SoC; ORB Feature Extraction; ROS; Computer Vision.

Resumo

Esta dissertação propõe uma nova arquitetura de acelerador para o algoritmo Oriented FAST and Rotated BRIEF (ORB), especialmente desenhada para Sistemas em Chip (SoCs) embutidos de baixa potência, respondendo às necessidades de eficiência computacional e energética na extracção de características de imagens em tempo real, no contexto de aplicações de visão por computador. O acelerador conta com uma arquitetura parametrizável, equilibrando a exatidão das características extraídas e a complexidade computacional, oferecendo compromissos específicos para cada aplicação. A arquitetura proposta foi também implementada num SoC FPGA embebido, integrando o acelerador desenvolvimento na lógica programável e implementando uma instância do Robot Operating System (ROS) em execução no processador, formando um dispositivo completo de extração de características de imagens. A integração desenvolvida com o ROS facilita a implementação em aplicações modulares de visão por computador.

Foi realizada uma avaliação experimental do acelerador proposto, utilizando imagens do conjunto de dados TUM-VI para avaliar a sua exatidão, robustez à rotação e eficiência energética. Os resultados obtidos mostram as vantagens da arquitetura de extração de características proposta, atingindo níveis de desempenho de outras soluções do estado-da-arte, obtendo ganhos de eficiência energética entre 6,7x e 16,2x quando comparado com as mesmas soluções.

Por fim, o dispositivo foi também validado com um conjunto de imagens reais de grutas subaquáticas numa demonstração em tempo real de sobreposição de características num vídeo transmitido para o dispositivo proposto, reforçando, assim, o potencial deste trabalho como solução viável para diversos conjuntos de aplicações.

Palavras Chave

Acelerador em Hardware; SoC Embebido; Extração de Características ORB; ROS; Visão por Computador.

Contents

List of Figures

List of Tables

Acronyms

2D	Two dimensional
3D	Three dimensional
AR	Augmented Reality
ASIC	Application-specific integrated circuit
AXI	Advanced eXtensible Interface
BBF	Best-Bin-First
BRAM	Block Random Access Memory
BRIEF	Binary Robust Independent Elementary Features
CPU	Central Processing Unit
DDC	Display Data Channel
DMA	Direct Memory Access
DOG	Difference-of-Gaussian
DRAM	Dynamic Random-Access Memory
DSP	Digital Signal Processor
FAST	Features from Accelerated and Segments Test
FIFO	First-In First-Out
FPGA	Field Programmable Gate Array
fps	Frames Per Second
GPIO	General Purpose Input/Output
GPU	Graphics Processing Unit
HDMI	High-Definition Multimedia Interface
HPC	High-Performance Computer
IMU	Inertial Measurement Unit

IP	Intelectual Property
KPI	Key Performance Indicator
LB	Line Buffer
LiDAR	Light Detection and Ranging
LUT	Look Up Table
MIPI CSI	Mobile Industry Processor Interface Camera Serial Interface
NMS	Non Maximum Suppression
ORB	Oriented FAST and Rotated BRIEF
PL	Programmable Logic
rBRIEF	Rotated Binary Robust Independent Elementary Features
RGB	Red Green Blue
ROV	Remotely Operated Vehicle
ROS	Robot Operating System
SD	Secure Digital
SfM	Structure-from-Motion
SIFT	Scale Invariant Feature Transform
SLAM	Simultaneous Location and Mapping
SoC	System on Chip
SSH	Secure Shell
SURF	Speeded Up Robust Features
TMDS	Transition-Minimized Differential Signalling
UAV	Unmanned Aerial Vehicle
USB	Universal Serial Bus
viSLAM	Visual Inertial Simultaneous Localization and Mapping
VO	Visual Odometry
WB	Window Buffer

1

Introduction

Contents

Image processing techniques are ubiquitous and have provided significant advances in many application areas. In Agriculture, for instance, it is already used to detect plant leaf diseases [6, 7] and in Medicine it is widely adopted for batch classification of graphical exams [8]. Specifically, the particular class of image feature extraction techniques is fundamental in robotics real-time applications such as Visual Odometry (VO) [9] and Simultaneous Location and Mapping (SLAM) [10–13] algorithms, making it possible to infer trajectories and maps of an agent's surroundings [14] and navigate through them [15]. These techniques are also key in Structure-from-Motion (SfM) techniques for 3D reconstruction frameworks, used to model unknown objects and environments, with applications in several fields including Archaeology [16], Geography [17], Medicine [18], and Geology [19].

In this work, feature extraction is particularly contextualised in the sensitive use case of underwater cave exploration. This activity is chosen for its importance for Geology [20], Hydrology [21], Biology [22], and Archaeology [16] studies. In fact, from an ecological perspective, groundwater is one of the main resources for human consumption and agricultural and industrial use. In Portugal, huge limestone massifs, such as Maciço Calcário Estremenho (MCE), are the main water supply for several municipalities, thus their preservation is critical and is achieved by tracking their changes through mapping and volumetric modelling. While presenting a great opportunity to deepen the understanding of the aforementioned scientific fields [23], the exploration of underwater caves coexists with logistical and technical challenges in navigating, documenting, and mapping those systems [24, 25]. As such, investment is being made to develop new and innovative tools to aid in their modelling with resource to computer vision based techniques [24, 25]. Following this trend, the 3D-CAVE project, for which this thesis contributes, proposes the development of a device capable of modelling underwater caves in real-time.

1.1 Motivation

Image feature extraction offers valuable information [6, 7, 9, 13, 14, 17–19], but often introduces significant challenges. An application where these benefits coexist with difficulties is the documenting of underwater caves that was traditionally done by divers who characterised these spaces using topography techniques through manual measurements of depth, distance, and azimuth [26, 27]. However, these methods were time-consuming and limited in the detail of the produced models. A modernisation of the field through the employment of technologies such as remote sensing (eg., LiDAR), photogrammetry, and other computer vision methods has enabled much faster and more detailed reconstruction procedures [28, 29]. Specifically, VO [9] and SLAM techniques can track the position of an observer (a cave diver in this scenario) and three-dimensionally map the surrounding environment. That being said, the deployment of these technologies in underwater caves is not straightforward, as caves are complex systems that can be long ($> 10km$) [23], have narrow passages and fragile geological formations that can

be easily damaged, requiring the presence of specially-trained divers and deeming the use of Remotely Operated Vehicles (ROVs) and Unmanned Aerial Vehicles (UAVs) unsuitable.

Currently, due to power and computational inefficiencies of the implementations of feature extraction algorithms, images are often acquired during a dive and only posteriorly processed in High-Performance Computers (HPCs) which achieves a dense Three dimensional (3D) reconstruction of the underwater cave systems [27]. This current workflow introduces great inefficiency to the documentation process, since the divers have no real-time feedback on the coverage and potential occlusions of the recorded footage. This results in an iterative process of diving and post-processing (see Figure 1.1). Providing the diver with real-time feedback on the mapping progress of the recordings allows for a more streamlined documentation process and ultimately faster modelling of the cave systems.

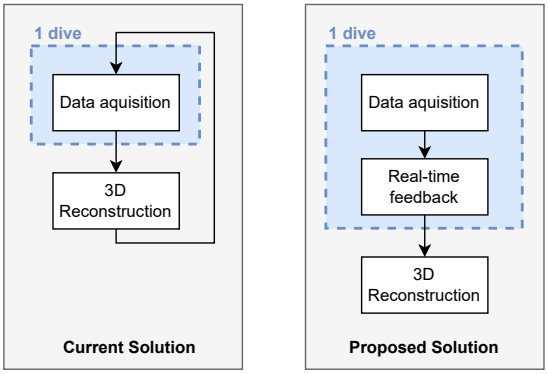


Figure 1.1: Current and proposed workflows of underwater cave mapping.

While underwater cave exploration is a clear example of the improvements still needed in computer vision applications, the mass adoption of image sensors in embedded systems [30] is a widespread concern in the most diverse application domains, which has brought the need for the development of efficient methods of processing such data. Considering that images are large blocks of data of which only specific regions offer relevant information, several popular techniques have been developed to extract these relevant features [1,2,31]. As a result, the introduction of feature extraction pre-processing is often a first step towards reducing the computational and energy requirements for the systems on which they are deployed.

Specifically, in the context of real-time embedded systems, this initial step often proves to be insufficient. Consequently, to improve both efficiency and time performance, innovative strategies are frequently employed. These strategies typically involve the acceleration of certain functional modules in dedicated hardware.

1.2 Objectives

The objective of this thesis is to examine the primary efficiency and performance constraints associated with the deployment of embedded real-time performing computer vision devices. This study aims to pinpoint specific functional components that could benefit from hardware acceleration, facilitating the deployment of portable, low-power, real-time performing devices across diverse application domains. In particular, these advances should be easily integrable in existing systems and, additionally, support the future creation of a system that maps underwater cave systems in real-time using video data. Thus, the thesis aims to:

- Study the building blocks of state-of-the-art image-based mapping algorithms;
- Identify which computer vision building blocks can potentially be hardware accelerated;
- Design dedicated hardware for the chosen blocks;
- Validate the work with commonly used datasets.

1.3 Contributions

According to the aforementioned objectives, a full hardware-accelerated feature extraction system was deployed to a low-power embedded System on Chip (SoC), targeted at real-time 3D mapping and navigation applications. Hence, the following contributions can be highlighted:

- A new efficient Oriented FAST and Rotated BRIEF (ORB) (feature extraction) accelerator architecture for low-power embedded SoCs, characterised by a configurable datapath and offering accuracy and complexity trade-offs for different applications;
- Accelerator performance metrics obtained through testing with images from a real underwater cave dataset;
- A complete feature extraction system, implemented on an embedded Field Programmable Gate Array (FPGA) SoC, comprising the proposed ORB accelerator deployed on the FPGA fabric and a software module running in the CPU;
- Deployment of the proposed ORB system as a complete Robot Operating System (ROS) [32] node, capable of obtaining image frames from an HDMI source input and publishing the extracted features as a ROS topic.

1.4 Outline

The remainder of this document is organised as follows. After this introductory chapter, Chapter 2 presents an insight to image-based state-of-the-art mapping solutions and the most relevant feature extraction algorithms, followed by a study of previous efforts towards accelerating them in dedicated hardware. Chapter 3 describes the accelerator designed for efficient feature extraction and is followed by details on its integration in a complete device that can be deployed to existing computer vision-dependent systems. Finally, Chapter 5 provides experimental validations on the described implementations. The document ends with a summary of the challenges and accomplishments that resulted from this thesis work and presents some future work guidelines and opportunities.

2

Background and State-of-the-art

Contents

With the ever-growing use of computer vision in the most diverse application domains, from industrial assembly lines [33] to autonomous driving [34] or satellite localisation [35] [36], there is a growing interest in solutions that deliver viable performance levels while meeting specific power and size constraints of the underlying platforms. Hardware accelerator devices, namely the ones implemented on Field Programmable Gate Arrays (FPGAs), are being used to fill this gap, and investments are being made to find new ways of leveraging them [37] [38].

The automotive industry, for example, is experiencing an increase in the complexity of its systems with the pursuit of autonomous driving and an increasing concern with power consumption to maximise driving autonomy. This has resulted in the research and implementation of innovative solutions using FPGAs [30] and while some applications require real-time treatment of the acquired sensor data because they control actuators in real-time [30], other systems such as star trackers take advantage of this real-time performance to avoid storing all acquired data [39]. The application of this sought-after technology in real-time applications with particular attention to power consumption further motivates its application in the context of underwater cave mapping. This mapping can be enhanced through the design of efficient dedicated hardware modules that implement specific elements of the mapping algorithms.

In this chapter, a context for image-based Three dimensional (3D) mapping is given followed by a description of state-of-the-art image feature extraction solutions. This is followed by an analysis of previous implementations of feature extraction accelerators, an introduction to FPGA platforms, and a discussion of the possible approaches to the problem at hand.

2.1 Image-Based 3D Mapping

Perceiving a 3D space from Two dimensional (2D) images is a classical photogrammetry problem that started to gain relevance in the 20^{th} century with some of its first relevant results in 1961 by Lawrence Gilman Roberts [40]. Photogrammetry initially focused on simply determining the depth of an object captured in a picture with methods analogous to human vision [41] but is later expanded to computer vision applications with Structure-from-Motion (SfM) [42], and Visual Odometry (VO) [9] methods which are later combined into Simultaneous Location and Mapping (SLAM) and other localisation and mapping algorithms [43].

A modern approach to SfM [42] establishes a pipeline for this type of algorithm composed of a correspondence search and a 3D reconstruction stage (Figure 2.1). The correspondence search stage is responsible for searching the input images for characteristic features, such as corners, and matching them between different images. The 3D reconstruction stage takes the matches of a specific feature and estimates its position, thus progressively constructing a 3D map of the captured environment. Visual

Odometry (VO) [9], on the other hand, focuses on the estimation of the trajectory of the camera. While SfM and VO have very similar architectures since both extract features from input frames and match them in order to obtain spatial information, VO is specifically designed for real-time applications and its output is a real-time camera pose estimate while SfM outputs a point cloud. These two methods represent two of the main building blocks for current state-of-the-art 3D mapping algorithms such as SLAM.

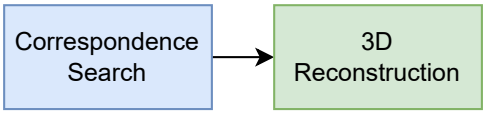


Figure 2.1: Structure from Motion pipeline.

2.1.1 Visual Simultaneous Localisation and Mapping

A common SLAM pipeline (Figure 2.2) starts with a search on frames captured by a visual sensor (input search) and often extracts specific features from these frames. This is followed by the Pose Tracking block that comprises the odometry calculations [43], updating the camera position using the current pixels/features, and the ones previously projected on a 3D space. The Mapping stage continuously places the detected features in the 3D space, and finally the Loop Closure module successively optimizes the positions of the features in that same space.

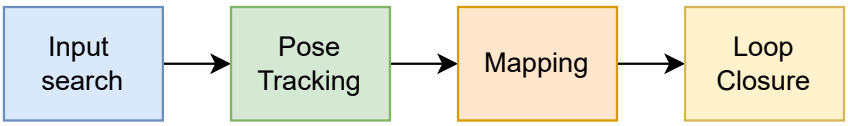


Figure 2.2: Common SLAM pipeline.

The Pose Tracking block requires previously positioned features on the 3D space and the Mapping block is responsible for doing so. This stage takes the newly identified features and determines their 3D position. While it is possible to do so directly from a single frame with a stereo camera setup, with monocular systems a feature matching between frames is required to determine the depth (location) of said feature. Visual Inertial Simultaneous Localization and Mapping (viSLAM) enhances the simpler visual SLAM pipeline through a sensor fusion between video and Inertial Measurement Unit (IMU) data to improve the accuracy and robustness.

The final block of the SLAM pipeline is precisely where the innovation of this family of algorithms resides. The Loop Closure feature of SLAM distinguishes it from classical VO and mapping algorithms, since it allows for the correction of the drift accumulated over time and propagates a correction backward on the estimated trajectory [43].

Considering the entire SLAM pipeline it is important to note that, the Input Search stage if often the most time consuming step of the entire pipeline, representing between 60 and 70% of the algorithm's runtime [44]. Being the computational and efficiency bottleneck of these image-based methods, the input search stage is considered a clear target for possible improvements.

2.1.2 Related Work

Mapping and localising an agent on unknown environment is a ubiquitous problem mainly in robotics [14, 34]. As such, SLAM algorithms are commonly deployed on embedded devices, some with higher performance and power requirements [34], and others with strict impositions due to size and energy availability [14,35]. The aforementioned algorithms have often been implemented on embedded Graphics Processing Units (GPUs) [45, 46], Central Processing Units (CPUs) [47, 48] and System on Chip (SoC) [14, 49] devices on board of Unmanned Aerial Vehicles (UAVs), humanoid robots and ground vehicles.

2.2 Feature Extraction

In a SLAM pipeline, the input search can be performed with direct methods (e.g., DTAM [12], ROVIO [50]) or indirect methods (e.g., MonoSLAM [10], ORB-SLAM [13]). Direct methods use all pixel information which is projected on a 3D space allowing for a more complete reconstruction of the captured environment [43]. These methods are more commonly used with stereo or multi-view camera setups which allow for the positioning of a pixel in a 3D space by using only the frame data. Since the most computationally intensive task of visual mapping systems is often the image processing stage [44], indirect methods, also known as feature extraction, are generally preferred for efficient embedded devices. These methods allow for a more efficient processing of the captured frames, as they extract characteristic features from the frames and only the corresponding regions are processed. While the direct and indirect specificity of the input search refers to the inputs of this block, the output can be respectively classified, according to Servières et al. [43], as either dense (e.g., DTAM [12], LSD-SLAM [11]) or sparse (e.g., ORB-SLAM [13], ROVIO [50]). Similarly to the input distinction, sparse methods mean that only the identified features are placed in the 3D space (point cloud), while dense methods output the entire frame after it has been projected in a 3D space (dense map). The combination of indirect/sparse methods is the most common approach to input search [43] and the most suitable for portable solutions, as it requires less computation and can be used with a monocular setup (one camera).

Considering the most prevalent indirect/sparse input search methods, the three main state-of-the-art feature extraction algorithms will be analysed in detail: SIFT [1], SURF [2], and ORB [31].

2.2.1 Scale Invariant Feature Transform (SIFT)

The Scale Invariant Feature Transform (SIFT) feature extraction algorithm is a breakthrough by Lowe [1] that advanced the scientific field of computer vision by introducing image feature descriptors that are invariant to scale and rotation [1]. This novelty allows for the development of reliable applications in the areas of image matching and object recognition. To achieve this, the SIFT algorithm steps through the following stages: scale-space extrema detection, keypoint localisation, orientation assignment, and feature descriptor construct (see Figure 2.3). The sequential flow of these steps allows for the algorithm to discard any potential keypoint at any stage and move to the next pixel which might allow for lower resource allocation depending on its implementation.

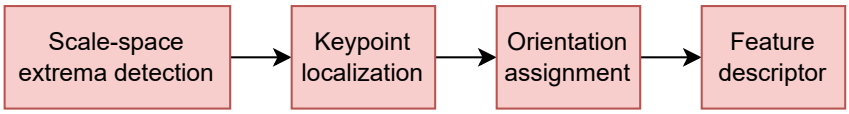


Figure 2.3: SIFT stages.

To understand the operation of the SIFT algorithm it is important to define the concept of scale-space (or octave) which will be referred to throughout the algorithm decomposition. In the context of SIFT, a scale-space is a set of images which are all the result of the convolution between an original image ($I(x, y)$) and variable-scale Gaussian ($G(x, y, \sigma)$) which results in a sequence of increasingly blurred images ($L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$). The scale of the Gaussians (σ) within a scale-space is spaced by a constant factor, k. An example of four scale-spaces, each with five levels/scales of blurring, is given in Figure 2.4 using one frame from an underwater cave data-set.

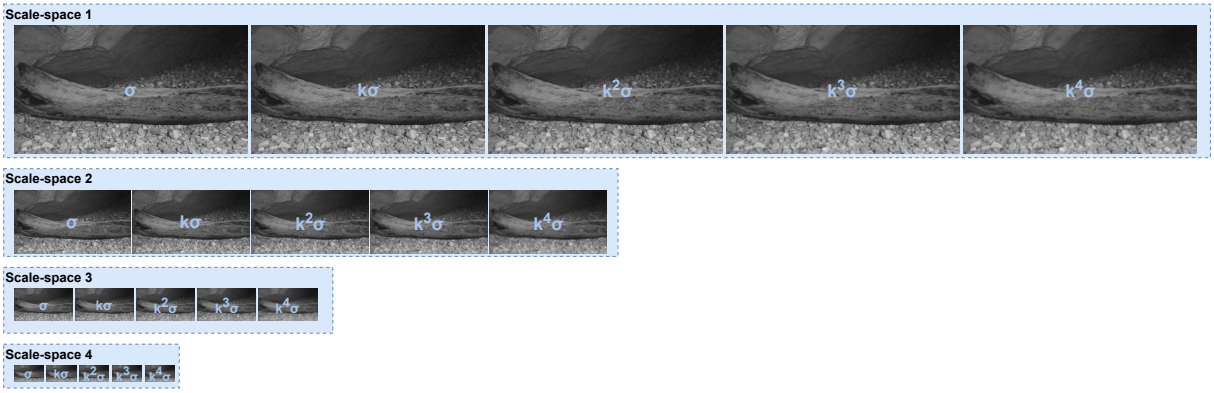


Figure 2.4: Illustration of four scale-spaces with five Gaussian scales each.

As depicted, multiple scale-spaces are generated for each frame and the base image for each octave is the result of the previous octave scaled by $1 : 2$. The previously mentioned stages of SIFT (shown in Figure 2.3) are executed for each scale-space (or octave).

Accordingly, the **scale-space extrema detection** step requires the computation of the difference of

consecutive scale-space images ($L(x, y, \sigma)$) named Difference-of-Gaussian (DOG). The DOGs are then used for selecting keypoints (pixels) which are either the maximum or the minimum within a $3 \times 3 \times 3$ region from the current and adjacent DOGs (see Figure 2.5). The local extrema check is followed by the **keypoint localisation** step which consists of checking if the potential keypoint has low contrast (making it susceptible to noise) or if it is poorly localized along an edge [1].

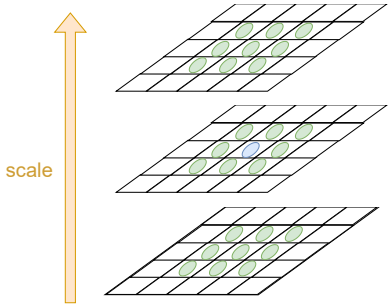


Figure 2.5: Extrema detection [1].

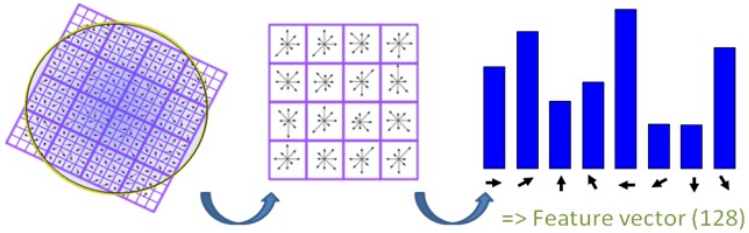


Figure 2.6: SIFT feature descriptor, sourced from [1].

Next, the **orientation assignment** stage starts the process of outputting a discovered feature (keypoint and surrounding region) by normalizing it. To do so, it starts by scaling the keypoint region according to the Gaussian scale where it was detected, followed by the assignment of the main orientation of the region. This main orientation is detected by computing the gradient at each pixel of the region and compiling all gradient orientations in a histogram of 8 orientations. The highest scoring bin of the histogram sets the main orientation of the feature which is rotated accordingly to make the future descriptor rotation invariant.

In the final stage of Lowe's algorithm [1], the feature descriptor is constructed. This construction starts with a 16×16 feature patch which is divided into a 4×4 matrix and for each elements, an orientation histogram is again computed. These histograms are then concatenated resulting in a vector of $4 \times 4 \times 8 = 128$ bins (Figure 2.6), composing the scale and rotation invariant descriptor, which is passed onto the proceeding SLAM stage.

2.2.2 Speeded Up Robust Features (SURF)

The Speeded Up Robust Features (SURF) algorithm, proposed by Herbert Bay et al. [2], introduces performance improvements to SIFT with the main goal of achieving real-time performance. The pipeline for SURF can be macroscopically represented as illustrated in Figure 2.3. The improvements introduced by SURF include optimising the stages previously established by SIFT and previous feature extraction algorithms.

The keypoint localisation in this algorithm is done by using the determinant of the Hessian matrix

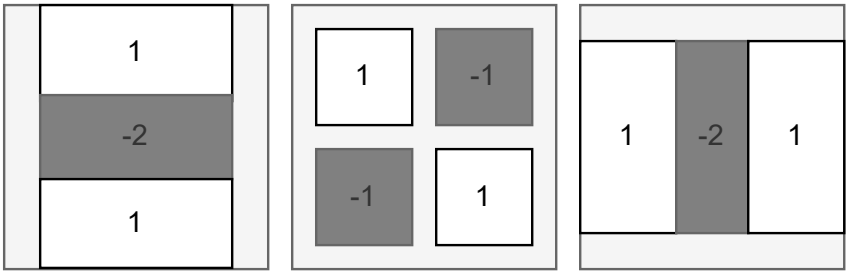


Figure 2.7: SURF simplified box filters [2].

for each point in an image. The Hessian matrix ($H(\mathbf{x}, \sigma)$, $\mathbf{x} = (x, y)$) can be constructed through the convolution of the image with the Gaussian second-order derivative. Bay et al. [2] further simplify Lowe's work [1] who approximated the Laplacian of Gaussian with a DOG, by constructing the Hessian matrix using integral images and simplified box filters (see Figure 2.7). The process of producing a blurred image is thus optimised time-wise.

Further optimisation is implemented by building the different scale-spaces with scale-varying filters instead of scale-varying images (as shown in Figure 2.8). This allows for the simultaneous/parallel determination of the multiple octaves instead of each octave depending on the previous smoothing. As in SIFT, the octaves are used to extract **local extrema** in a $3 \times 3 \times 3$ region. To reject poorly **localized keypoints**, the maxima of the determinant of the Hessian matrix are used and interpolated in scale and image space. This detection algorithm is named **Fast-Hessian** [2].

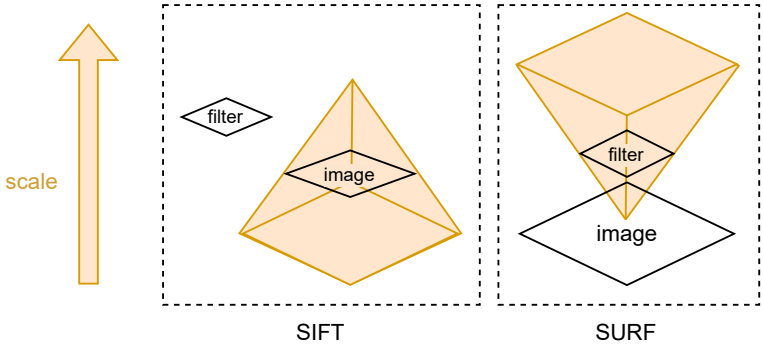


Figure 2.8: SURF scale-space construction.

Although acknowledging the good performance of the SIFT descriptors, Bay et al. [2] propose a complexity strip-down of Lowe's approach [1] to ease computation on both descriptor construct and matching.

The descriptor construct is comprised of the **orientation assignment** and the determination of the components of the descriptor. For the descriptor to be invariant with rotation, a reproducible orientation for the feature is identified.

While SIFT required the calculation of the tangent inverse, SURF uses the Haar-wavelet responses in the x ($H_x = [-1, 1]$) and y ($H_y[-1, 1]^T$) directions with a kernel size dependent on the scale at which the keypoint was found. The responses are then weighted with a scale-adapted Gaussian ($\sigma = 2.5s$) and represented as vectors in a space where the abscissas are the horizontal responses and the ordinates the vertical responses. The orientation of the feature is then extracted through the sum of the responses within a rotating window of $\pi/3$ radians and the selection of the resulting dominant orientation (see Figure 2.9). The first step to the construct of the **descriptor** is the placement of a squared region of size $20s$

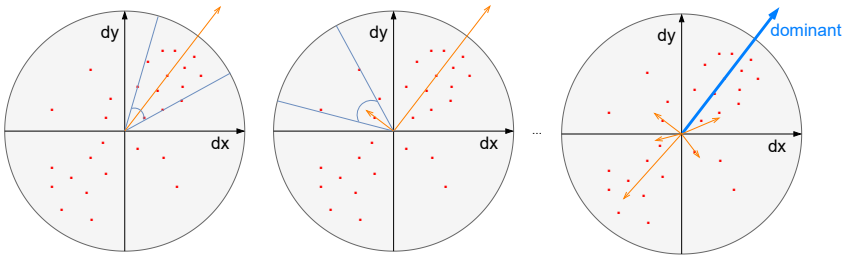


Figure 2.9: SURF orientation assignment.

centred on the keypoint and oriented according to the previously determined dominant orientation. This region is then divided into 4×4 smaller subregions, and for each, the Haar-wavelet responses are once again calculated for 5×5 regularly spaced sample points. The responses in both x and y directions (d_x, d_y) are then summed, as well as their absolute value, resulting in a four-dimensional vector describing that subregion ($\mathbf{v} = \{\Sigma d_x, \Sigma d_y, \Sigma|d_x|, \Sigma|d_y|\}$). Concatenating all of these vectors adds up to a 64-bit feature descriptor. It is also possible to divide the feature into 3×3 subregions which output a 36-bit descriptor instead. This SURF-36 approach is poorer at describing a feature but allows for faster matching [2].

2.2.3 Oriented FAST and Rotated BRIEF (ORB)

The Oriented FAST and Rotated BRIEF (ORB) [31] algorithm has the goal of providing an optimised alternative to SIFT and SURF while also having the major advantage of not being protected by intellectual property. The steps described by this algorithm are illustrated in Figure 2.10 and it is designed on the basis of two well-established techniques: Features from Accelerated and Segments Test (FAST) [51] for detecting keypoints and Binary Robust Independent Elementary Features (BRIEF) [52] to construct its descriptors.

FAST Corner Detector

Contrary to the previously analysed algorithms, ORB does not start by applying different levels of blurring to an image within the same scale, rather it starts by detecting the FAST corners/features of the image.

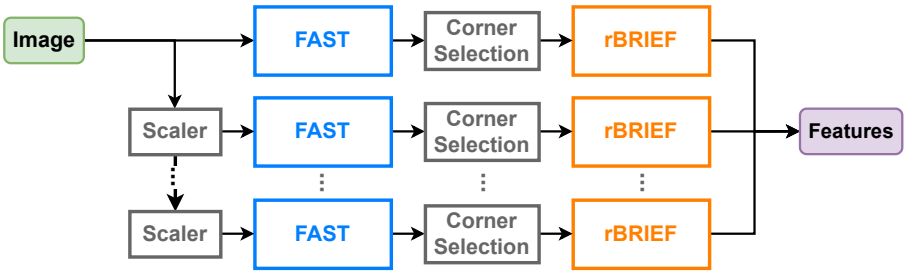


Figure 2.10: ORB algorithm pipeline.

FAST tests a circumference of diameter 7 centred on pixel i of an image to check if it is a corner (see Figure 2.11). For the 16 pixels that make up that circumference, it checks if they are similar, brighter, or darker than i and counts how many fit each classification. If more than 8 consecutive pixels are brighter or darker than i, the detector considers it a corner, i.e., a point of interest for the following stages [53]. Though being very non-complex, this feature detector has no measure of cornerness, making it too responsive along edges. As such, Rubilee et al. [31] found the need to pick only the top responses of the FAST detector thus requiring a measure of curviness to order the potential keypoints. They use a Harris corner measure [54] to do so which allows for the picking of only the K-top corners found for a given image. Up to this stage no scale invariance has been guaranteed and this is handled by applying this FAST and Harris filtering to a pyramid of sequentially down-sampled images, making it possible to achieve a level of scale invariance.

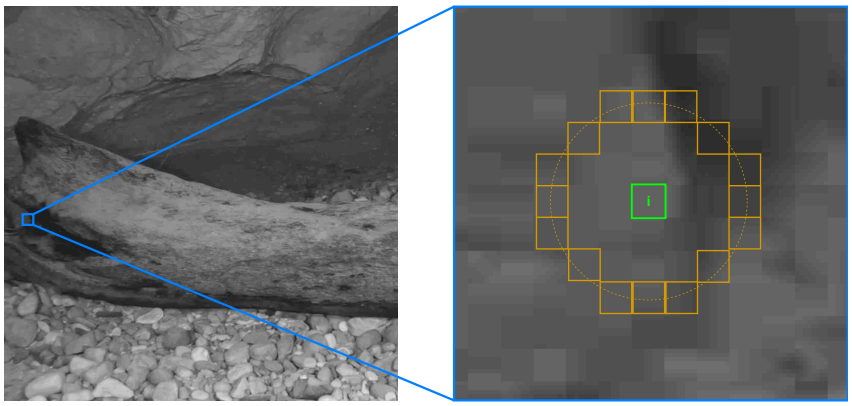


Figure 2.11: FAST corner detector, radius of 3.

It is important to note that this does not necessarily introduce a serial component to the algorithm as the different scale images are obtained simultaneously from the original image, as opposed to SIFT which requires the images from previous octaves to construct the downsized ones.

Rotated BRIEF descriptor

With the keypoints identified, ORB, as the previously described algorithms, starts the construction of the feature descriptor by determining its orientation. It does so using an intensity centroid (C) [31] [55]. Since the corners associated with each feature (detected by FAST) are offset from the feature itself, the process of finding the intensity centroid of the corner consists of determining the moments of a smoothed feature region $\mathbf{p}$ (of 31×31 pixels), centred on pixel i, according to Equation 2.1, where $\mathbf{p}(x, y)$ is the intensity/luminance of the pixel at (x, y) coordinates of the blurred image. The intensity centroid is calculated from the moments values, from Equation 2.2. The orientation of the feature is then given by the vector from the centre of the feature to its centroid ($\theta = atan2(m_{01}, m_{10})$).

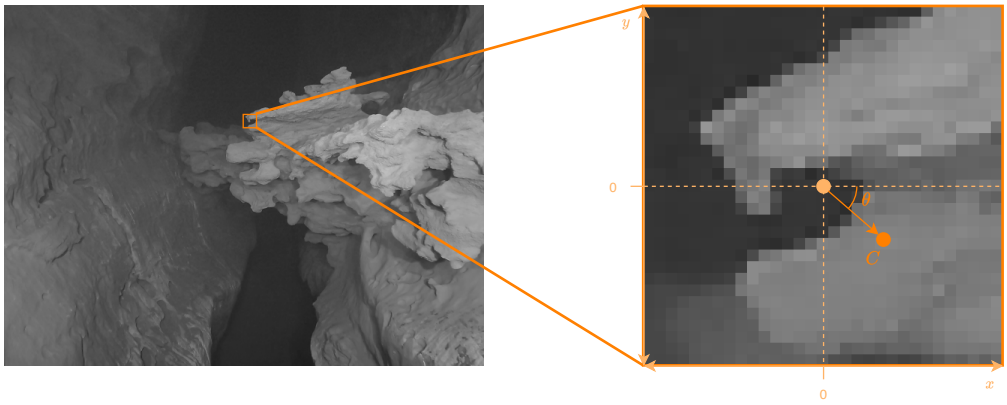


Figure 2.12: rBRIEF intensity centroid.

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \tag{2.1}$$

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}} \right) \tag{2.2}$$

The assignment of an orientation to the feature (or patch as named by Rublee et al. [31]) enables the construction of Rotated Binary Robust Independent Elementary Features (rBRIEF) descriptor that builds on the BRIEF descriptor by Calonder et al. [52]. To do so, pairs of points ($\mathbf{x}$ and $\mathbf{y}$) are given by a predefined pattern ($\mathbf{S}$)(see Figure 2.13) with low correlation between pairs that maximises success and minimises false positives when matching the descriptors [44].

$$\tau(\mathbf{p}, \mathbf{x}, \mathbf{y}) \begin{cases} 1, & \mathbf{p}(\mathbf{x}) < \mathbf{p}(\mathbf{y}) \\ 0, & \mathbf{p}(\mathbf{x}) \geq \mathbf{p}(\mathbf{y}) \end{cases} \tag{2.3}$$

$$f(p) := \sum_{1 \leq i \leq 256} 2^{i-1} \tau(\mathbf{p}, \mathbf{x_i}, \mathbf{y_i}) \tag{2.4}$$

To make the descriptor rotation invariant, the pair of points is rotated according to the already assigned orientation, and these updated point coordinate pairs ($\mathbf{S}_\theta$) are used to run the binary tests $\tau(\mathbf{p}, \mathbf{x}, \mathbf{y})$ (Equation 2.3), where $\mathbf{p}$ is the smoothed feature region. The feature descriptor is then con-

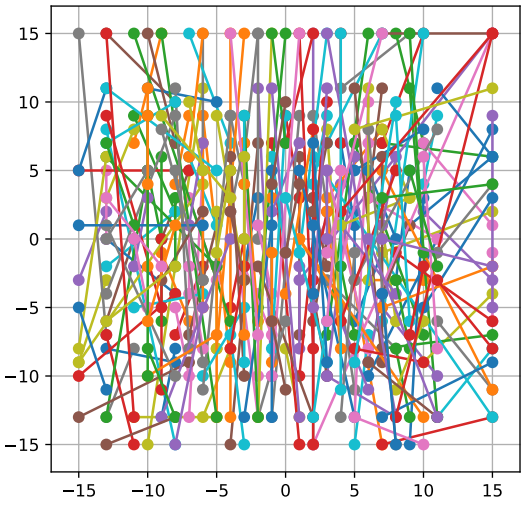


Figure 2.13: Example of a BRIEF pattern ($\mathbf{S}$).

structed as a vector of 256 binary tests (Equation 2.4). Rublee et al. settled on a 256-bit descriptor [31].

2.2.4 Feature matching approaches

To extract spatial information from frames with different views of an object it is necessary to match the recognized features between views. Each descriptor produced by the previously detailed methods proposes a matching strategy that best suits its characteristics.

SIFT and SURF propose the use of a nearest neighbour strategy to match detected features against a database of previously seen features [1, 2]. The nearest neighbour is chosen utilizing the minimum Euclidean distance between descriptors. The distance between the nearest neighbour and the second nearest neighbour so it does not solely check the closest match, rather it provides an estimate of the density of false matches and checks if the nearest neighbour is a strong match. For large databases, which often exist for large maps, it is important to have optimized methods for finding neighbours. In particular, the SIFT algorithm uses an approximate algorithm by Beis and Lowe [56] named Best-Bin-First (BBF) which returns the closest neighbour with high probability. Similarly, for SURF, Herbert Bay et al. [2] matches its descriptors with the nearest-neighbour strategy but also with the similarity threshold technique. The latter matches according to a threshold on the Euclidean distance of the feature to the single nearest neighbour on the database.

Whereas SIFT and SURF produce descriptors that represent the orientations within a feature with a sequence of numbers, ORB's descriptor is composed by a series of binary tests, thus the matching

between these descriptors, while still done through the nearest-neighbour strategy, uses the Hamming distance instead of the Euclidean distance between neighbours [31]. In this application, the Hamming distance counts the number of different bits between two descriptors which is less computationally demanding than the Euclidean distance that require multiplications.

2.3 Accelerating Feature Extraction

Despite contributing to make localisation and mapping systems more efficient [43], feature extraction is often still the main computationally demanding and energy consuming task in their pipeline. Consequently, several efforts have been made to accelerate corner detection [14] and feature extraction with dedicated hardware. Specifically, ORB has been widely adopted for these systems since it was originally intended to provide reliable and efficient feature extraction [14,44,57–59]. Many of these solutions target FPGA devices due to their flexibility and fast deployment features.

2.3.1 Overview on FPGA Platforms

FPGAs are commonly used for their low-latency, low-power, and reconfigurability [30]. These are integrated circuits that contain logic and memory cells that can be reconfigured and reconnected as needed. They allow the development of digital systems that achieve a level of parallelization that would require several common processor cores while being easily reconfigurable, making them suitable for production or development environments. While CPUs are particularly good for the fast development of general applications, this flexibility and abstraction level coexist with overheads that make them inefficient. FPGAs on the other hand, require much more time-consuming development phases since the designs are done at a hardware level. However, this means that there is less processing overhead in the final solution, making them much more efficient (see Figure 2.14) [30].

Furthermore, operating at a bit level is particularly interesting when working with video feeds. While in a CPU there is a layer of abstraction that limits the access of applications to complete video frames, with FPGAs it is possible to access a video stream at the hardware level, meaning that partial frame information can be processed as soon as it is transmitted.

Systems that make use of these devices fit into two main categories: loosely and tightly coupled. FPGAs can be used stand-alone and only interact with other devices with common communication protocols such as PCIe, UART, and I2C among others. While simple applications can benefit from this type of device, systems like the ones envisaged in this thesis benefit from a different architecture. In particular, tightly coupled systems such as SoCs combine Field Programmable Gate Arrays (FPGAs) with CPUs in a single chip. This allows for much better integration since the less complex tasks can be implemented in software, and more demanding tasks are implemented of the FPGA while making use of

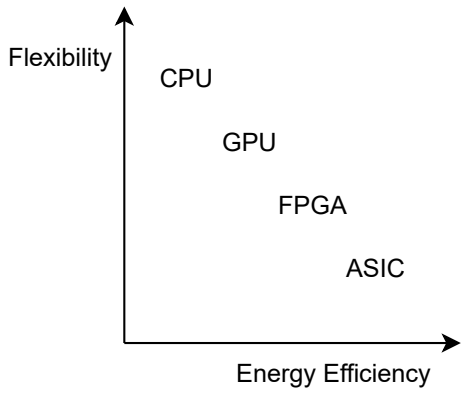


Figure 2.14: FPGA efficiency comparison.

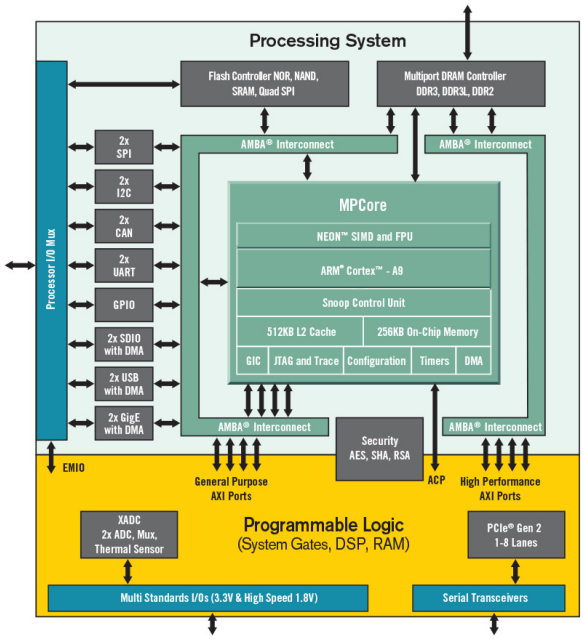


Figure 2.15: Tightly coupled SoC, sourced from [3].

the fast interfaces between these two components (FPGA↔CPU). The coupling between these devices is usually done by means of shared memory space and specific interaction blocks (eg: Advanced eXtensible Interface (AXI)-General Purpose Input/Output (GPIO)). An example of a tightly coupled device is shown in Figure 2.15.

2.3.2 Hardware-based ORB accelerator

An initial contribution to the acceleration of ORB was made by Janosch Nikolic et al. [14] who implemented a hardware module to accelerate the FAST feature detector on a low-cost, low-power device, significantly reducing the computational complexity of their SLAM pipeline and enabling its employment on resource-constrained platforms. Fang et al. [57] and Vibhakar Vemulapati et al. [44] pushed the efficiency of ORB even further, proposing a complete ORB hardware module to extract features from 640×480 pixels images at around 60 Frames Per Second (fps), thus achieving real-time performance, at a power consumption lower than that of the ORB implementation in software.

In real-time applications, images are often transmitted one pixel per clock cycle [60]. As such, both [14] and [57] adopt streaming strategies that consist of pushing a single pixel to the ORB accelerator on each clock cycle. This is particularly interesting for embedded real-time devices, as it allows pixels to be processed as soon as they are captured and contours the need for storing entire frames in hardware, enabling low-latency and fewer hardware resource usage. Mateusz Wasala et al. [60] and Runze Liu et al. [4] make two other propositions that use this pixel streaming strategy and implement complete ORB

accelerators. Both works use the RS-BRIEF descriptor introduced by [4], that uses a symmetric test pattern composed of consecutively rotated coordinate pairs (see Figure 2.16), thus making the rotation invariance of the extracted descriptor achievable by rotating the descriptor rather than rotating the test pattern. This saves several hardware resources since only one pattern has to be saved and the rotation task involves simple bit shifts. However, when originally proposed [4], the authors recognise that this technique is likely to produce descriptors that do not perform as well as the rBRIEF regarding orientation and matching. The fact that they also fail to provide specific metrics for rotation invariance makes it use undesirable in an early stage of development.

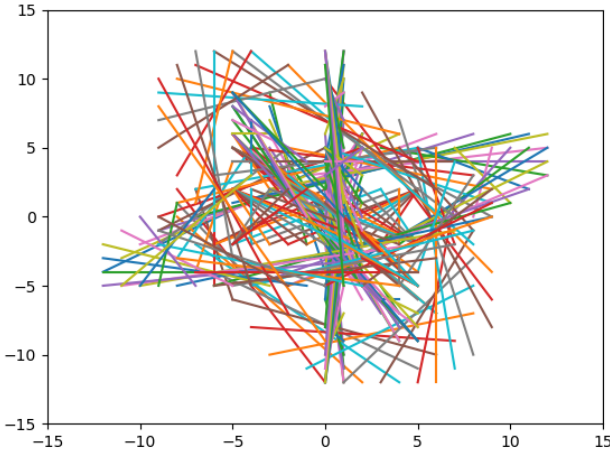


Figure 2.16: RS-BRIEF pattern composed of several rotations of 3 coordinate pairs [4].

In Table 5.3 a comparison between the performance and energy efficiency achieved by these contributions when compared to ORB implementations in software executed both in a desktop CPU (Intel Core i5) [57] and on an embedded GPU (Nvidia Jetson Nano) [61].

Table 2.1: ORB Accelerators performance comparison in terms of throughput and energy efficiency.

Work	Device	Resolution	Latency [ms]	Throughput [fps]	Power [mW]	Energy Eff. [mJ/frame]
[60]	XCZU7EV	3840x2160	#NP	60	5042	84
[44]	XCZU3EG	640x480	2.5	62	+4600	+74
[57]	Altera Stratix V	640x480	14.8	67	4556	68
[4]	XCZ7045	640x480	9.1	55.87	1936	35
[57]	Intel Core i5	640x480	25	40	16000	400
[61]	Nvidia Jetson Nano	640x480	#NP	45	+3484	+264

+ Energy consumption not provided for isolated ORB accelerator. # Not provided.

Despite encouraging the possibility of attaining a real-time implementation of an ORB accelerator, these works target higher-end devices (Intel Altera Stratix V FPGA [14], XCZU3EG [57], and XCZU7EV [60]), with the exception of [4] who targeted the lower-end XCZ7045 SoC. Furthermore, all four efforts are still above the power consumption constraints that often characterise embedded systems (< 1W).

This is particularly critical when light, portable, battery-powered devices are the target.

2.3.3 System Integration

Commonly in the computer vision domain, feature extraction is the starting point for complex tasks. The works mentioned previously [4, 44, 57, 60] developed ORB accelerators to improve the time performance and energy efficiency of complete SLAM pipelines using SoCs. As already discussed, the visual processing of these pipelines is their main bottleneck, as such the ORB stage is the only module accelerated by these teams in hardware, with the remaining stages (pose tracking, mapping, and loop closure) implemented in the SoC's CPU.

Although the use of SoCs allows for the complete integration of computer vision applications in a single device, many existing, more complex, and modular systems would benefit from using the referred ORB accelerators [30, 38]. This implicates the integration of standard interfaces on these devices as to make them easily adoptable.

2.4 State-of-the-art Discussion

Several approaches to the problem of extracting spatial information from visual data were analysed. Let us discuss the main topics covered as to define the best references for the accelerator that shall be designed.

Feature extraction

Considering the presented feature extraction methods and their potential hardware acceleration, it is clear that ORB is one of the best candidates [44, 57, 60]. This interpretation is supported by several previous solutions that chose ORB as the reference algorithm for their hardware designs implemented in FPGA platforms. They accomplished not only real-time performance but managed to greatly reduce energy consumption (up to x11.4 lower than the CPU implementation). These works also set target values for the latency and throughput of a feature extraction device. Considering the use case, here explored of underwater cave mapping, an analysis is conducted to find a requirement for latency and throughput.

The ease of use of a mapping device is for once dependent on the delay between an action by the user and the resulting feedback on the device, as studied by Ming Li et al. [62]. This study analysed the impact of image processing latency and display refresh rate on the ease of use of pointing an Augmented Reality (AR) device to a target. This is similar to the work at hand, since a diver will have to point a device to objects that have not yet been correctly mapped. Based on the findings of Ming Li

et al. and previous work done on the implementation of similar solutions by Stephen Se et al. [63] a minimum refresh rate of 30Hz and latency of 60ms should make for a fluid display of feedback [62] with unperceived delay [64]. Refinement of the refresh rate is usually necessary once a complete system is in place, since, as concluded by Ming Li et al., the ease of use is not a function of only the refresh rate or latency, but rather a function of both and a too high of a refresh rate for a specific latency can potentially worsen the experience of the user.

Despite meeting the aforementioned timing requirements, the existing solutions target higher-end devices that are costly, and have a power consumption above the values often target by embedded, battery-powered devices.

System integration

Being commonly one of the many stages that compose computer vision systems, it is relevant to consider how a newly designed ORB accelerator will integrate these larger, modular systems. Although current devices offer single-package solutions, this strategy limits the potential integration of the designed ORB accelerators in the many other application domains for which they are suited.

Proposed contributions

Significant progress has been achieved in the development of an energy-efficient device capable of real-time ORB feature extraction. Nonetheless, the current solutions still fall short of providing comprehensive feature extraction functionalities suited for low-end hardware. Additionally, they do not yet meet the demanding energy efficiency requirements necessary for devices powered by portable batteries. Finally, while contributing with innovative accelerator designs, the mentioned works failed to develop the interfaces required to integrate their devices in a wider range of applications. Taking into account the increasing demand for efficient image processing systems, there is a clear lack of a device that achieves the requirements already mentioned while also being easily integrable in a variety of complete systems.

2.5 Summary

In this chapter a review of the following thematics was presented: image-based 3D mapping, feature extraction methods, and the hardware acceleration of these methods. Bearing in mind the multiple disciplines where computer vision is one of the main innovation propellants, the focus is steered to underwater cave mapping as a use case, starting with a brief description of the initial methods used for documenting these systems. SfM and SLAM are then introduced as state-of-the-art techniques currently employed for this purpose, along with the presentation of their main building blocks. Of the 4 macro modules identified for SLAM (see Figure 2.2), image processing (or input search) is identified as the

main computational and efficiency bottleneck, leading to the introduction of feature extraction methods as an alternative to methods that use complete images for spatial awareness. Three state-of-the-art feature extraction algorithms (SIFT, SURF, and ORB) are then described in detail and a conclusion is made that ORB is the most suitable for efficient embedded applications.

The state-of-the-art review ends with a survey on previous attempts to accelerate ORB with dedicated hardware leveraging FPGA technologies integrated in tightly coupled SoCs that allow for complete computer vision applications to be implemented, namely SLAM pipelines.

3

ORB Accelerator

Contents

This chapter presents a computationally and energy-efficient accelerator architecture for the Oriented FAST and Rotated BRIEF (ORB) [31] algorithm, described in VHDL, and particularly tailored for embedded system deployment. As such, it considers the previously stated constraints, namely, a power consumption below 1 W, a latency of less than 60 ms, and a throughput of 60 fps. The proposed ORB [31] accelerator takes as an input greyscale images, which are assumed to be streamed at a rate of one pixel per clock cycle (as elaborated in 3.1), and outputs detected features alongside their Rotated Binary Robust Independent Elementary Features (rBRIEF) descriptors. It integrates a dedicated data orchestration infrastructure and the ORB feature extraction module, consisting of a Features from Accelerated and Segments Test (FAST) [51] corner detector, which identifies 7×7-pixel regions of interest (features) within the input image, and a rBRIEF [31] descriptor constructor, which constructs 256-bit descriptors (refer to Figure 3.3). These descriptors can then be used to match features across different images, when the accelerator is integrated on a larger computer system.

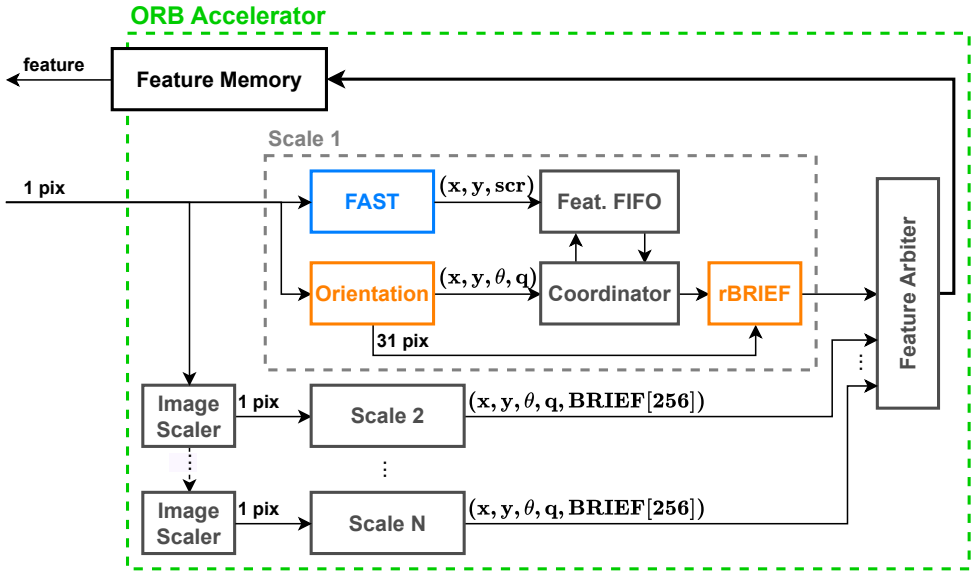


Figure 3.1: ORB accelerator design overview.

3.1 Data Orchestration

In real-time applications, images are often transmitted one pixel per clock cycle [60]. This happens not only because image sensors produce frames at this rate but also because this strategy minimises processing latency. In fact, common interfaces, such as High-Definition Multimedia Interface (HDMI) or Mobile Industry Processor Interface Camera Serial Interface (MIPI CSI), often stream image frames with such rates (1 pixel / clock cycle). As such, the proposed ORB accelerator was designed to receive one pixel each clock cycle, also ensuring that at no moment are the contents of an entire frame stored in

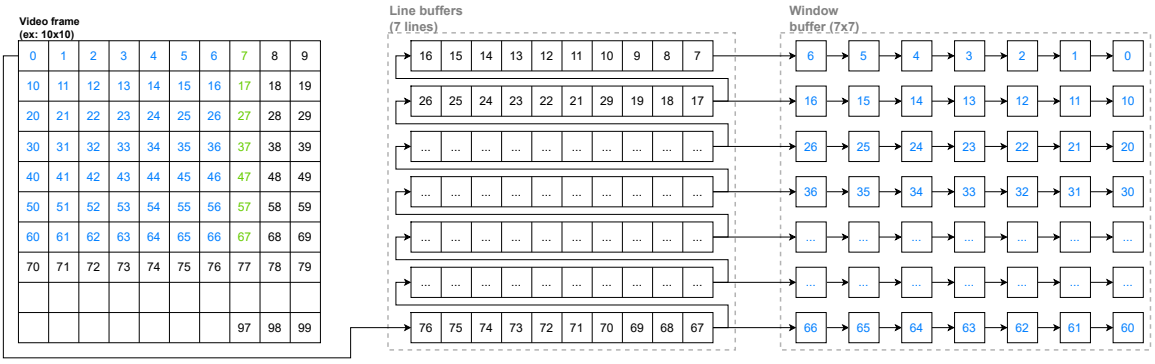


Figure 3.2: Line and Window buffer example.

hardware, thus reducing memory resource requirements.

Hence, the accelerator's data management infrastructure is composed of two main hardware modules, necessary for input and intermediate (partial) image buffering and output feature storage. These structures are described in detail below.

3.1.1 Input Image Buffering

The ORB algorithm uses several sliding windows on the input images to detect relevant features and construct their descriptors. This nature of the algorithm, alongside the pixel streaming architecture used, requires dedicated image buffering to hold the values of each transmitted image line. Accordingly, several Line Buffers (LBs) and Window Buffers (WBs), implemented with 8-bit shift registers (see Figure 3.2), are placed inside several modules (see Figures 3.4 and 3.7). The window buffers are populated with the output of each line buffer, reproducing a sliding window over the buffered image, as can be seen in Figure 3.2. Naturally the memory resources used by the modules will be directly dependent on the line dimension of the input image. Also, latency is introduced to the accelerator since a sliding window (WB) is only valid once the number of lines received is greater than the height of said window. However, this allows for a significant reduction of necessary memory resources, which are often quite limited in low-end hardware platforms.

3.1.2 Feature Memory

As a result of the considered pixel / cycle input rate, only a single feature is extracted per clock cycle by the ORB accelerator. Moreover, since ORB [31] extracts features from several scales of the input image (as was detailed in Chapter 2.2.3), which means that more than one feature can be made available at each instant, implying the need for data buffering and selection mechanisms. These are done with a dedicated arbiter module (see Figure 3.3) that implements a queue for the extracted features and is

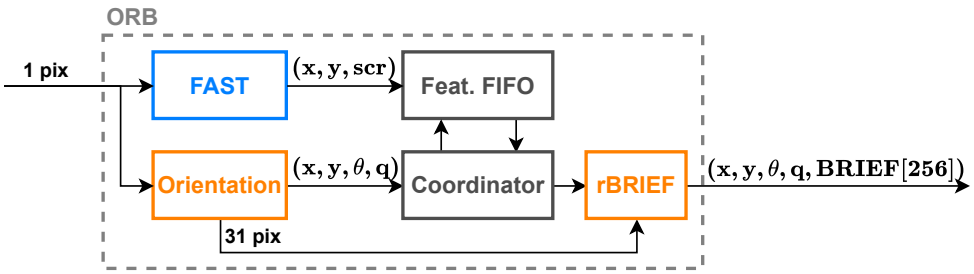


Figure 3.3: Single scale ORB module.

responsible for writing the queued features to a memory structure that is externally accessible from the accelerator's peripherals.

3.2 ORB Feature Extraction Module

The ORB module of the accelerator is composed of two main components: the FAST corner detector, and the rBRIEF descriptor construct. Both modules are detailed in the following sections.

As previously detailed in Chapter 2, the FAST [51] algorithm detects features by comparing the luminance of the 16 pixels on a circumference of a 7-pixel diameter, with the luminance of the centre pixel. If more than 9 contiguous pixels on the circumference are either darker or brighter than the centre (i.e., the difference is lower/higher than the negative/positive threshold), the region is considered to be a corner/feature. To construct an rBRIEF descriptor, 256 coordinate pairs are used, whose composition is encoded in 256 bits. The comparison between the pixels of each pair is done within a 31×31 pixels smoothed test region ($\mathbf{p}$) centred on a detected feature. Hence, for each pair, the corresponding bit on the descriptor is 1 if luminance of the first pixel is lower than the second, and 0 otherwise.

For the constructed descriptors to be invariant to the orientation at which a feature is detected, ORB introduced the rotated Binary Robust Independent Elementary Features (BRIEF) algorithm. It computes the main orientation of a feature region (31×31 pixels), and rotates the coordinates of the pixels to be compared accordingly, using the latter for the construction of the final descriptor (as also detailed in Chapter 2). The orientation, Θ, of the image patch is determined through its luminance momentum on the x (m_{10}) and y (m_{01}) directions (see Equation 2.1) where x and y are the pixel horizontal and vertical coordinates from the 31×31 pixel region ($x, y \in [-15, 15]$). Lastly, the scale invariance is achieved by applying the aforementioned steps to consecutively down-scaled images.

3.2.1 FAST Corner Detector Module

The FAST [51] corner detector implementation involves three main steps: corner detection, corner score computation, and feature selection (see Figure 3.4).

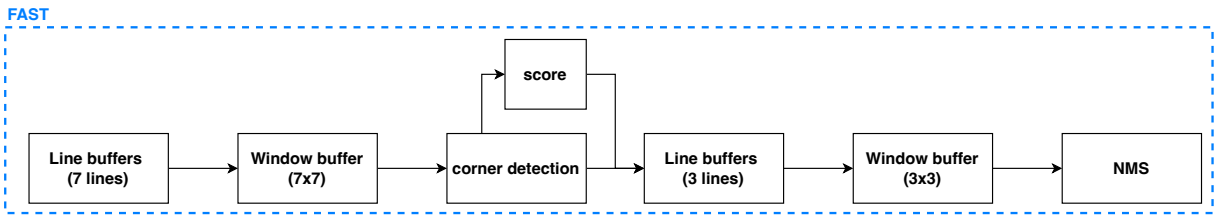


Figure 3.4: FAST corner detector submodule overview.

The corner detection module (see Figure 3.4) processes each block of 7×7 pixels and computes the luminance comparisons for the 16 tested pixels (using the luminance thresholds that can be configured externally). These comparisons are stored in two 16-bit-wide vectors (for brighter and darker results) and checked (with AND logical operations) against bitmasks of the 16 different possible locations of 9 consecutive brighter/darker pixels. If at least one of the 32 bitmasks is matched, a corner is dentified. This logic and data flow are represented in Figure 3.5. While the corner detection module is checking for brighter/darker contiguous pixels, the resulting differences are also used to compute the corner score as the sum of their absolute values. The higher the sum, the stronger the corner is considered.

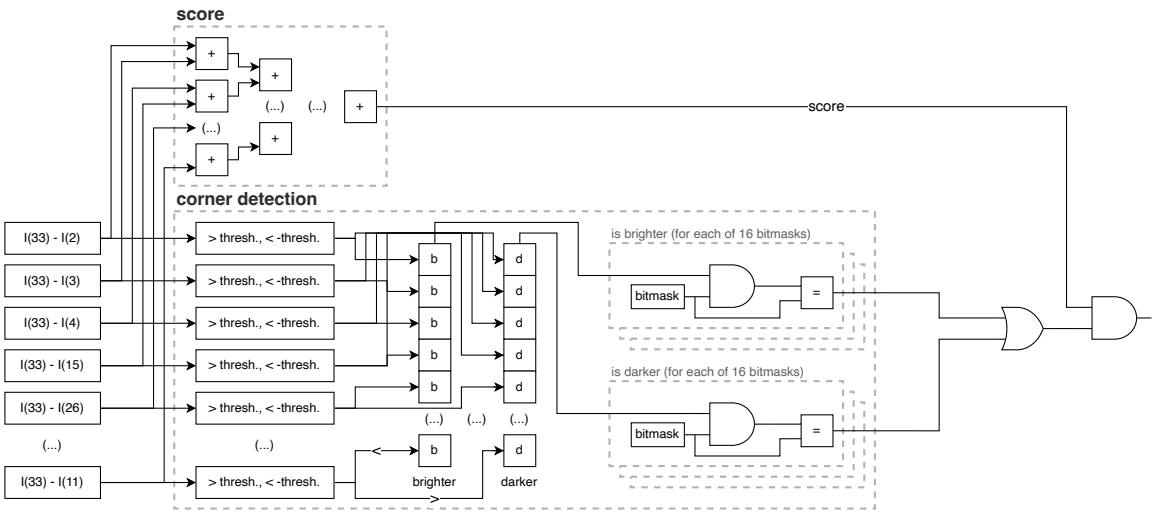


Figure 3.5: FAST corner detection.

The original ORB [31] algorithm proposes a global selection to pick the highest scoring features of all detected ones. Although providing a better selection of features per frame, this strategy would imply sorting the detected features, which would further increase the latency and computational resource usage of the proposed accelerator. Considering that the aim of this project is a low-latency and resource-constrained scenario, a local 3×3 region Non Maximum Suppression (NMS) is employed. If the pixels in the FAST sliding window (WB) are considered to be a corner, this score is passed to the NMS stage which, using, 3 LBs and a WB, will discard weak corners.

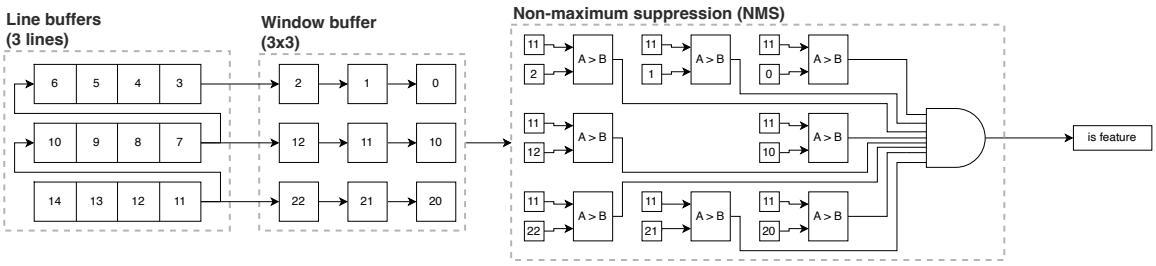


Figure 3.6: Non-Maximum Supression (NMS) module architecture.

This module is designed to be scalable with parametrizable width and height of the input image. As previously observed, the resource utilisation of the module depends solely on the length of lines in the input image.

3.2.2 rBRIEF Descriptor Constructor Module

The implementation of the rBRIEF [31] algorithm in hardware consists of three main steps: image smoothing, feature orientation computation, and the constructor of the rBRIEF descriptor (see Figure 3.7).

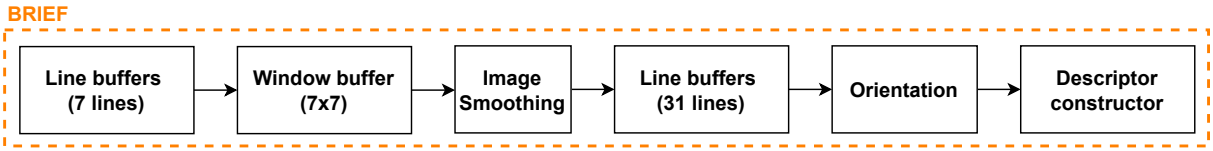


Figure 3.7: rBRIEF module overview.

Image Smoothing

A Gaussian filter is used to smooth the image before computing the feature orientation [31]. Instead of using a regular Gaussian matrix which would require several computational resources, an adaptation is made so that the convolution is done by simpler constant multiplicand operations. This module convolves the binomial Gaussian matrix ($\sigma = 2$) [44] in Equation (3.1) with a 7×7-pixel image patch stored in a WB.

$$G = \begin{bmatrix} 1 & 6 & 15 & 20 & 15 & 6 & 1 \\ 6 & 36 & 90 & 120 & 90 & 36 & 6 \\ 15 & 90 & 225 & 300 & 225 & 90 & 15 \\ 20 & 120 & 300 & 400 & 300 & 120 & 20 \\ 15 & 90 & 225 & 300 & 225 & 90 & 15 \\ 6 & 36 & 90 & 120 & 90 & 36 & 6 \\ 1 & 6 & 15 & 20 & 15 & 6 & 1 \end{bmatrix} << 12 \tag{3.1}$$

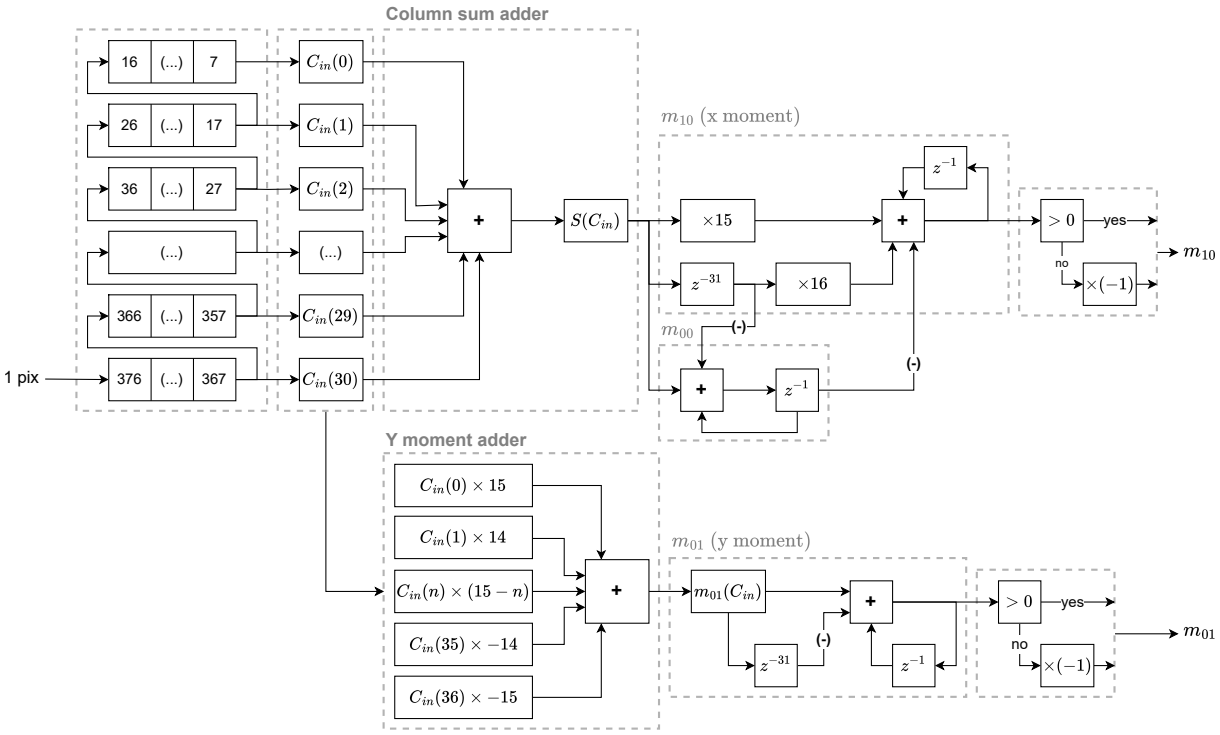


Figure 3.8: Image momentum computation diagram.

Orientation

The computation of the orientation of a feature patch (31×31 pixels) would intuitively require 31 LBs and a 31×31 pixels WB, since it requires the weighing of all pixels within said region [31]. Considering the 8 bit representation of the pixel luminance, it would be necessary to store close to 1 kB of data and perform several computations per orientation. This, otherwise resource intensive task, is reformulated in such a way that it is only necessary to store the first and last columns of the sliding window at each instant [44], using shift registers. This greatly reduces the use of computational and memory resources.

The orientation of the image patch (θ) is given by the vector that connects its intensity centroid (C) [31] (Equation 2.2) to the centre of the patch ($\vec{OC}$). Since the window slides across the x axis the y dimension does not change. This means that the y luminance momentum (m_{01}) can be determined as:

$$m_{01_{n+1}} = m_{01}(C_{in}) + m_{01_n} - m_{01}(C_{out}), \tag{3.2}$$

where C_{in} and C_{out} correspond to the elements of the incoming and outgoing columns respectively. Likewise, the computation of m_{00} also results from the addition and subtraction of the incoming/outgoing column's contributions:

$$m_{00_{n+1}} = S(C_{in}) + m_{00_n} - S(C_{out}), \tag{3.3}$$

where $m_{00_0} = 0$, and $S(C_{in})$ and $S(C_{out})$ are the sum of the incoming/outgoing column's.

Conversely, calculating the x-momentum (m_{10}) must account for the contribution of each column changes with the slide of the window. Assuming the storage of the entire 31×31-pixel window, the x-momentum is given by:

$$m_{10} = \sum_{x,y=-15}^{15} x\mathbf{p}(x,y) \tag{3.4}$$

Furthermore, considering that the right edge of the 31×31 window has a weight of 15 (see Figure 2.12), the contribution of the incoming column is given by:

$$m_{10}(C_{in}) = 15 \sum_{y=-15}^{15} \mathbf{p}(15,y) = 15S(C_{in}) \tag{3.5}$$

Similarly it is possible to obtain the outgoing column's contribution as:

$$m_{10}(C_{out}) = -15 \sum_{y=-15}^{15} \mathbf{p}(-15,y) = -15S(C_{out}) \tag{3.6}$$

Moreover, every time the window slides 1 pixel to the right, the momentum contribution of each column decreases by one (see Figure 2.12). This means that the momentum of each column can effectively be determined as $m_{10_{n+1}}(C_n) = m_{10_n}(C_n) - S(C_n)$. With this in mind, the patch luminance momentum in the x direction can be computed as:

$$\begin{aligned} & m_{10_{n+1}} = m_{10_n} + m_{10}(C_{in}) - m_{10}(C_{out}) - (m_{00_n} - S(C_{out})) \Leftrightarrow \\ \Leftrightarrow \quad & m_{10_{n+1}} = m_{10_n} + 15S(C_{in}) + 16S(C_{out}) - m_{00_n} \end{aligned} \tag{3.7}$$

The data flow of the computations mentioned above is illustrated in Figure 3.8. Having the x and y momentums, the patch orientation is calculated by $\Theta = arctan(m_{01}/m_{10})$. This would require doing arc-tangent operations which are very computationally demanding. Instead, by dividing each quadrant of the trigonometric circle into a parametrizable number (N) of discrete sectors (see Figure 3.10), it is possible to simplify the angle determination to only a few multiplications with constant values and comparisons. Since $|m_{10}| \times tan(\theta) = |m_{01}|$, a priority encoder is used to find the closest match for θ (see Figure 3.9). This priority encoder makes the comparison from Equation 3.8 for each of the N possible angles (see Figure 3.9). The multiplication of m_{10} by the N constant tangent values was simplified as an adder and shift tree using the Spiral Multiplier Block Generator [65].

Lastly, the quadrant can also be easily determined by checking the signal of both the x and the y momentums.

$$|m_{10}| \times tan(\theta) > |m_{01}| \tag{3.8}$$

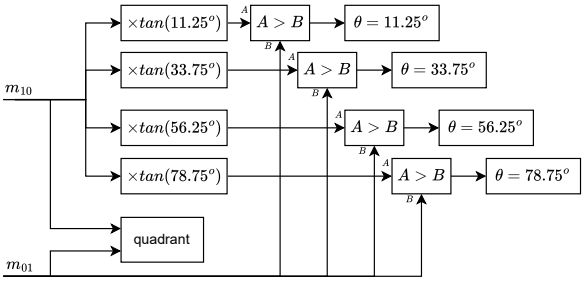


Figure 3.9: Orientation priority encoder.

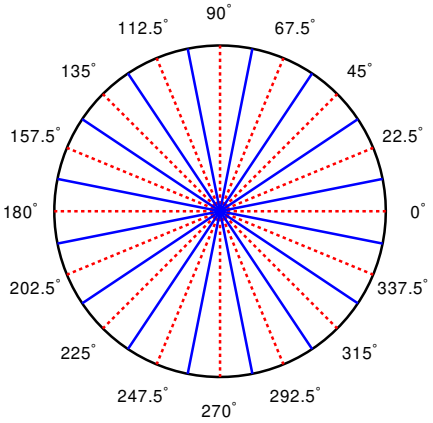
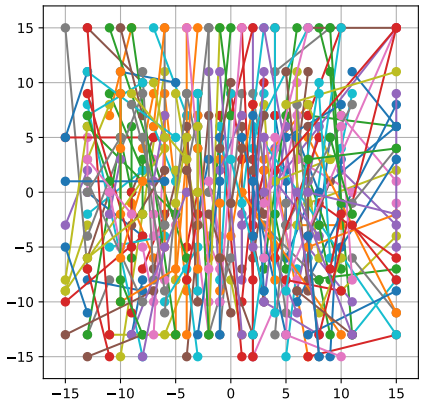


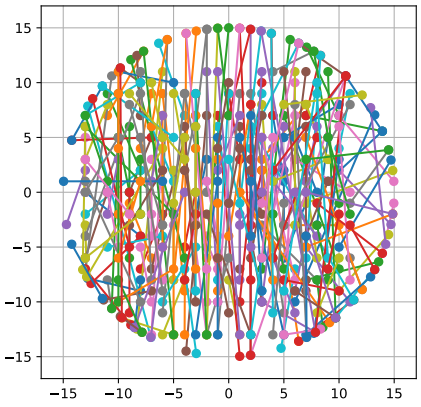
Figure 3.10: Orientation θ discretised in 16 sectors or $N = 4$.

Descriptor Constructor

The rBRIEF constructor module operates on 31×31 elements, making 256 comparisons according to the rBRIEF patterns. Rotating a square 31×31-pixel BRIEF pattern would imply the need for a 37×37-pixel WB. To make the module more memory efficient, the coordinates of the pairs considered are limited to a circle with 31 pixels in diameter (instead of a 31×31 square region) so that the rotated coordinates never exceed an absolute value of 31 making the required WB smaller (see Figure 3.11).



(a) 31×31 square pattern.



(b) Circular pattern with diameter 31.

Figure 3.11: Original and proposed rBRIEF patterns.

Furthermore, to avoid the need for a large multiplexing structure, this window buffer is implemented using structured memory elements addressable in 32-bit lines. Hence, by using 4 dual-port memories with 32-bit wide words it is possible to write/read the 31 elements (8 bits each) of a region's column

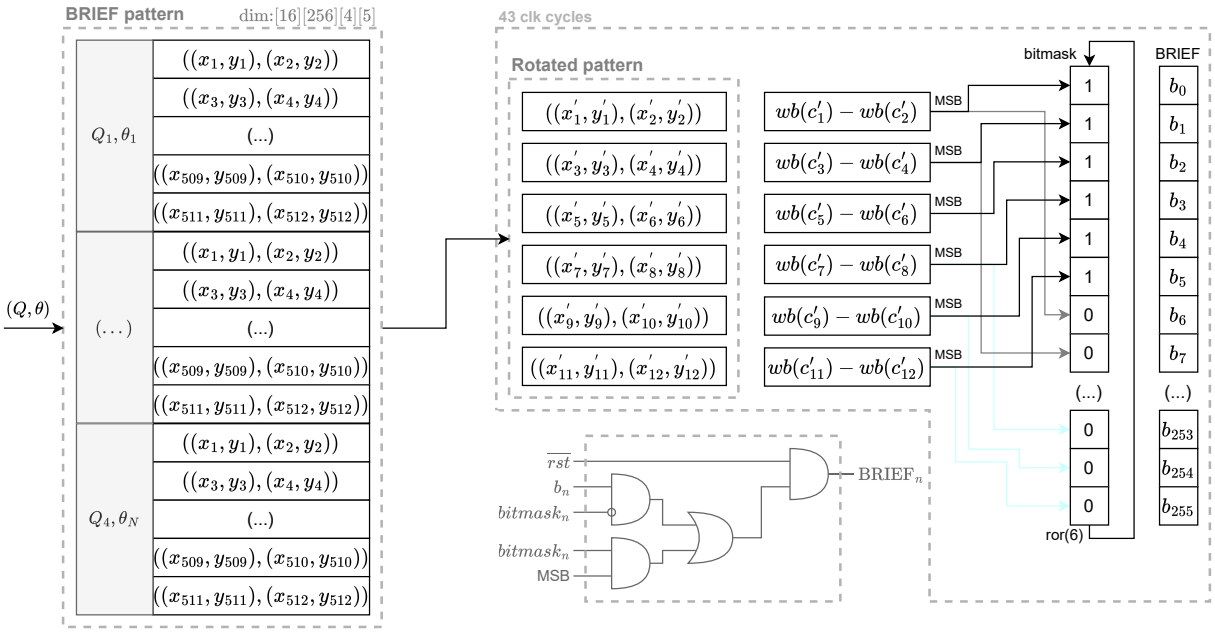


Figure 3.12: rBRIEF descriptor construct.

per clock cycle. This window is replicated 3 times to access 6 different pixels per clock cycle. As such, 3 pairs of pixels are read per clock cycle, resulting in 86 cycles to construct the descriptor and during this period new matches between the orientation and FAST First-In First-Out (FIFO) are discarded. The incremental construction of the descriptor is done with the help of a rotating bitmask (see Figure 3.12). This strategy trades off slightly increased latency of the construction of descriptors, with a significant reduction in resource usage.

Finally, since rotating 6 coordinates per clock cycle would require a considerable amount of hardware resources, the simplification proposed in [31] is followed to precompute the rotated patterns using Equation 3.9. The base addresses for accessing the memory containing these rotated patterns are composed as a function of the quadrant and orientation, and their size will inevitably grow with the increase of the number of sectors chosen for the orientation.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{3.9}$$

3.2.3 Image Scaler

As described in the explanation of ORB, the scale invariance of this algorithm is achieved by feeding successively downscaled images to modules identical to the ones just described. These image scalers are represented in Figure 3.3, and are implemented by means of a simple averaging down scaler with a

$2:1$ scaling factor within a 2×2 sliding window. This scaling process is exemplified in Figure 3.13. The number of scales is parametrizable according the application scenario.

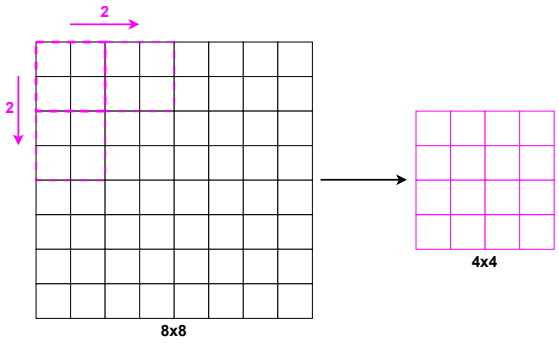


Figure 3.13: Example of the Scaling Box Filter

3.2.4 Summary

In this Chapter, it is detailed the proposed ORB accelerator architecture. It employs a pixel streaming architecture that not only ensures minimal latency but also reduces the necessary memory hardware resources, as the input image is stored only partially at any time. This module consists of three primary components, each undergoing distinct optimisations through either algorithmic relaxations or latency compromises.

The FAST algorithm is implemented according to the guidelines in its reference publication, with configurable contrast thresholds. The first optimisation occurs during the selection of the top-scoring detected features: instead of selecting the highest-scoring features across the entire frame, maximums in a local region of 3×3 pixels are chosen. Significant relaxation in the accelerator's overall design is introduced in the orientation determination stage, where the number of orientation sectors is discretised into a finite and adjustable number, allowing users to balance accuracy and efficiency at implementation time. The specific effects of this relaxation will be discussed in Chapter 5. Finally, a straightforward, yet crucial design decision, involves decomposing the descriptor construction over multiple clock cycles, substantially lowering hardware resource usage for a minor increase in descriptor construction latency.

The next Chapter explains how this module is integrated on a complete device than enables it to be easily integrated in larger systems that benefit and are made possible through such an efficient feature extraction module as the one here presented.

4

FPGA-Accelerated ORB Feature Extraction System

Contents

Efficient feature extraction devices hold significant potential to enable numerous novel computer vision applications [6, 38, 66], particularly in contexts where low power consumption and minimal latency are of paramount importance [4, 14]. These devices must possess the capability to be directly interfaced with image sensors [14], thereby not only simplifying the system's architecture but also minimizing latency between data acquisition and the subsequent output of detected features to the remaining stages of their specific applications [4, 44, 57, 60].

Contemporary robotic systems typically employ not only image-capturing modules but also a myriad of other devices that augment vision and perception capabilities (e.g., Light Detection and Ranging (LiDAR), Graphics Processing Units (GPUs), Inertial Measurement Units (IMUs), etc.), as well as devices that facilitate actuation on their environment (e.g., electric motors, pneumatic actuators, etc.) [38, 67]. For several years, these complex modular systems have necessitated a standardized communication protocol that allows orderly and straightforward communication [32]. Although each application could potentially adopt its own communication standard, Robot Operating System (ROS) [32] has increasingly been embraced by robotics system developers and manufacturers.

Following the development of a comprehensive Oriented FAST and Rotated BRIEF (ORB) accelerator architecture, a step further is taken by enabling its integration into a full ROS system. This is done not only targeting the mapping use case that is here considered, but also opening the accelerator's integration for alternative applications. This chapter presents the proposed ORB acceleration system implemented on an Field Programmable Gate Array (FPGA) System on Chip (SoC), which is interfaced with a video source and transmits the detected features on the frames via Ethernet or other available platform interfaces. In addition, the designated device operates a ROS node that facilitates both the transmission of the detected features and the reception of configuration commands to and from devices within a larger ROS network.

4.1 FPGA-based Feature Extraction

The proposed ORB accelerator was implemented as part of a low-power embedded feature extraction system designed to allow seamless integration into larger systems. It was deployed on a Digilent Zybo Z7-20 board equipped with an XC7Z020 SoC (xc7z020clg400-1) (Central Processing Unit (CPU) + FPGA). The FPGA Programmable Logic (PL) houses the ORB accelerator and the modules necessary to acquire the frame sequence from an High-Definition Multimedia Interface (HDMI) video stream, while the embedded ARM CPU configures the HDMI interface and host a ROS node that configures the accelerator and reports the extracted features to a ROS-capable system [32]. The complete high-level architecture of this device is presented in Figure 4.1.

The Digilent Zybo Z7-20 board integrates the necessary hardware components to power the SoC,

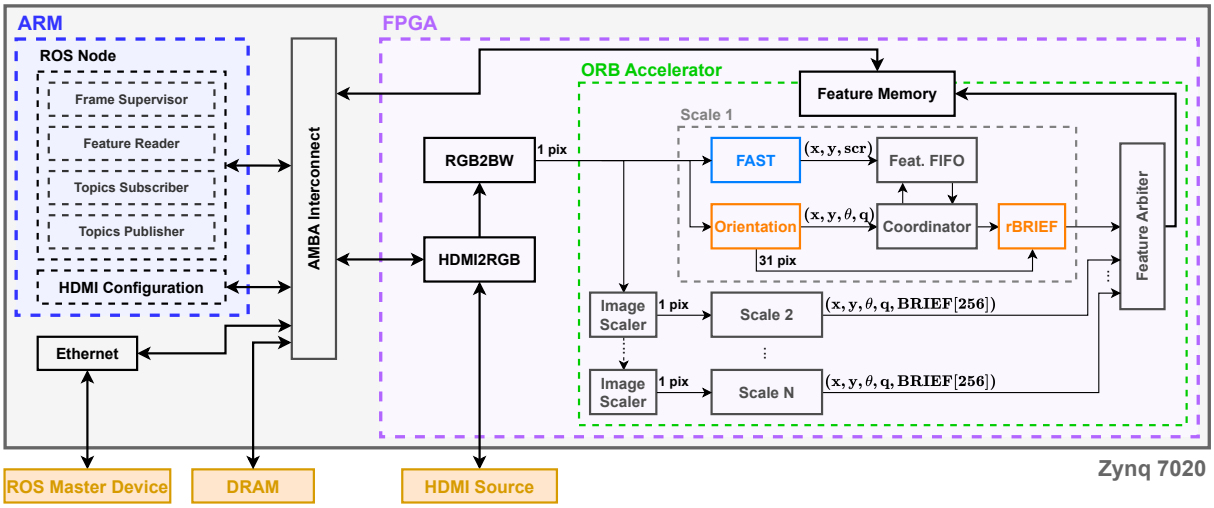


Figure 4.1: Complete ORB accelerator system with all the major functionality blocks and interfaces between the SoC CPU and FPGA.

handle HDMI signal reception and transmission, and connect to an Ethernet network. It carries the XC7Z020 SoC (xc7z020clg400-1), which boasts a Xilinx Artix-7 FPGA device alongside a dual-core ARM Cortex-A9 CPU capable of operating at frequencies up to 866 MHz. The CPU is coupled with a 1 GB Dynamic Random-Access Memory (DRAM). The system contains 4 32-bit Master/Slave Advanced eXtensible Interface (AXI) channels that interface with the PL fabric, as well as 4 AXI 64-bit/32-bit Memory channels. Additionally, the SoC is equipped with dedicated Direct Memory Access (DMA) channels to interface with Universal Serial Bus (USB), Ethernet, and Secure Digital (SD) peripherals. The available FPGA resources are summarised in Table 4.1. The selection of this device was influenced by its available interfaces, cost-effectiveness, low power consumption, and prior adoption in related projects [14].

Table 4.1: Zybo Z7-20 specifications.

	PS	Interface	PL			
Max. Freq. [MHz]	**DRAM [GiB]**	**DMA Channels**	**Logic Cells**	**LUTs**	**Block RAM [Mb]**	**DSP**
866	1	8	85K	53,200	4.9	220

4.2 ORB Accelerator FPGA Integration

To implement the proposed ORB accelerator on the FPGA fabric, the input video signal requires preprocessing, the ORB modules should be configured according to the use case at hand and the device to which it is synthesised, and the features produced must be accessible by the CPU.

4.2.1 ORB Accelerator Parametrisation

As previously discussed, the ORB accelerator is designed with scalability and performance trade-offs that can be tuned through the following parameters: horizontal and vertical resolution, Features from Accelerated and Segments Test (FAST) [51] contrast threshold, number of orientation sectors, and number of image scales. Specifically, the increase of the horizontal resolution of the input images increases the number of memory elements since the line buffers will increase in length, in contrast the change of the FAST contrast threshold does not impact the resource usage but will result on more or fewer detected features which will impact the number of constructed descriptors and ultimately the power consumption of the module. Accordingly, the number of sectors influences the robustness of the descriptors to rotation (this correlation is further discussed in Chapter 5), and the higher the sector count the higher the number of precomputed Binary Robust Independent Elementary Features (BRIEF) patterns, thus increasing the amount of memory resources used. Lastly, the number of scales affects the scale invariance and the number of computational and memory resources used.

Accordingly, for the adopted low-end FPGA, the ORB accelerator was configured for images with a resolution of 640×480 pixels (also keeping up with other state-of-the-art implementations [4, 44, 57]), a contrast threshold configured by the CPU according to the processed scene, 32 orientation sectors, and 2 scales.

4.2.2 Image streaming

As described in Chapter 3 and as seen in Figure 4.1, the ORB accelerator accepts a stream of 1 pixel per clock cycle. Hence, the accelerator is connected to a video stream from the board HDMI video input. This architecture has two important benefits: any computer or conventional camera can be connected to the accelerator through HDMI and stream a data-set to it just like a regular monitor or a live recording to integrate the device in a functional computer vision application. The use of the HDMI interface implies the negotiation between the source of the stream (external device) and the sink (SoC), and the translation of the decoded 24-bit Red Green Blue (RGB) to 8-bit luminance pixels, detailed below.

HDMI Decoding

Communication via HDMI entails the utilisation of three Transition-Minimized Differential Signalling (TMDS) signal pairs, which are responsible for the transmission of pixel data and frame synchronisation signals. Additionally, there is a Display Data Channel (DDC) that adopts the I^2C protocol to facilitate the transmission of specifications for both the source and the sink. Developing these protocols directly in hardware poses significant complexity; therefore,an Intelectual Property (IP) module provided by Digilent is employed, as depicted in Figure 4.1 under the HDMI2RGB label. This module not only engages in resolution

negotiation with the video source, but also decomposes the image stream into three fundamental signals: a 32-bit RGB pixel, as well as vertical and horizontal frame synchronisation signals. As illustrated in Figure 4.2, the HDMI protocol incorporates vsync (vertical) and hsync (horizontal) synchronisation signals essential to track the exact position of the pixel at any given moment. Hence, these synchronisation signals are crucial for generating a real-time overlay of detected features onto an HDMI output during live demonstrations. This being said, the accelerator does not employ these signals to establish its location on the frame. Instead, it relies on a pixel count.

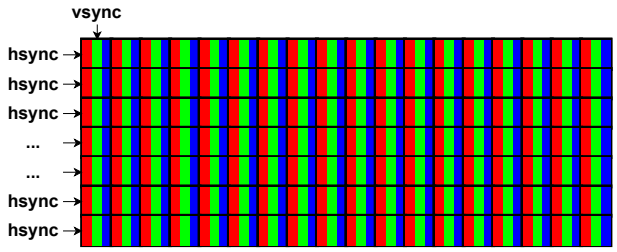


Figure 4.2: HDMI vertical and horizontal synchronization signals.

RGB to Luminance Conversion

Since the ORB [31] algorithm operates on greyscale images, specifically 8-bit luminance pixels. To facilitate this, a specialised hardware module, RGB2BW, was conceptualised as shown in Figure 4.3. This module performs a straightforward conversion between colour formats by computing the contribution of each colour component to the overall luminance using the formula: $0.2126 \times R + 0.0722 \times G + 0.7152 \times B = lum$. This transformation was simplified as an adder and shift tree using the spiral multiplier block generator [65] achieving a 7 clock cycles latency which is considered negligible for the overall accelerator.

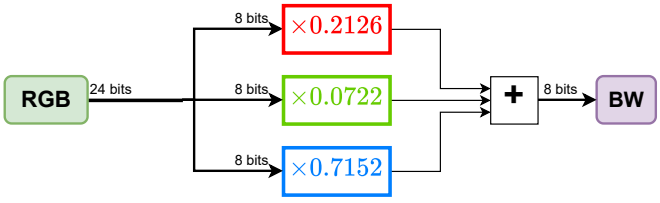


Figure 4.3: RGB2BW submodule architecture.

4.3 Accelerator Management and ROS Integration

Integrating the ORB accelerator with the CPU enables efficient data transfer between the feature extraction module and software applications. This step is key for the configuration, supervision, and man-

agement of the accelerator, enabling smooth operation within, for example, large robotic systems. Such integration is critical for the modular and adaptable nature of contemporary robotics, ensuring standardised interaction among various components.

4.3.1 Robot Operating System Overview

In the domain of robotics, systems tend to have a complexity associated with the multidisciplinary nature of the area, as well as the demanding requirements they impose [32, 38]. Modern robotics systems typically involve a vast array of sensors, perception and odometry modules, interface components, visualisation tools, and actuators [38, 68]. Associated with the diversity of components that make up these robots is also a diverse set of programming languages and types of implementations (e.g., CPU, GPU, FPGA, Application-specific integrated circuit (ASIC), etc). Orchestrating communication in an orderly and rapid fashion between all of these subsystems is not trivial and was a common challenge faced by academia and industry alike [69]. Several frameworks were proposed to solve this problem, but they were often targeted to specific applications and focused on perceived issues in each specific domain. The ROS [32] project is a widely adopted framework for handling communication in large modular robotics systems with five highlighted characteristics: peer-to-peer architecture, multi-language support, tools-based, reliance on widely used libraries, and is open-sourced.

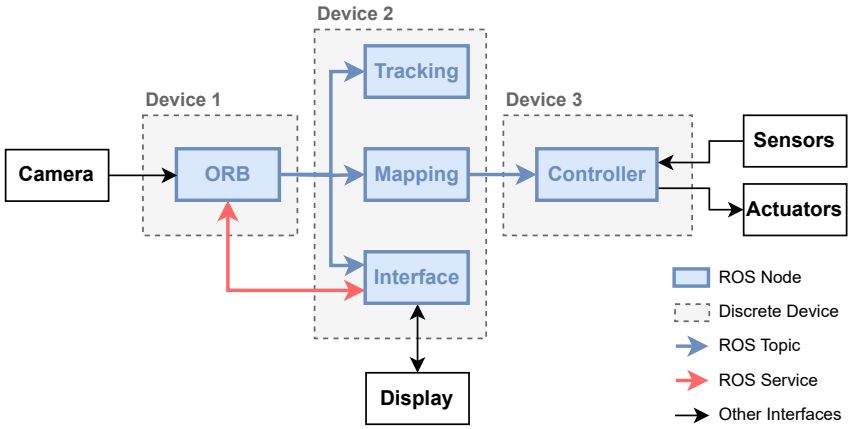


Figure 4.4: Example of a distributed robot architecture that uses a ROS for data interchange between its modules.

A ROS system's core component is the ROS node. Nodes typically handle a singular function within a larger modular system, and they interact with other nodes (tasks) by publishing/retrieving information through messages categorised under distinct topics. Each topic is unique, yet they may share message types. If a node wishes to subscribe to a topic, it simply needs to be on the same network as the publisher. Another form of data exchange is services, which operate on a request-reply basis, unlike the publish-subscribe model, providing utility for operations necessitating holding data transfers. ROS employs widely accepted libraries for compilation and standardized files to describe topics, messages,

and services, ensuring these descriptions are cross-platform. Thus, nodes can be compiled for various devices, facilitating the design of modular systems utilising diverse device topologies for specific tasks.

Figure 4.4 illustrates an example modular robot employing ROS for managing communication among functional blocks. In this scenario, the proposed ORB accelerator would be responsible for processing images captured by a camera, extracting features, and publishing them to a features topic. The tracking, mapping, and interface components would subscribe to this topic to track the robot's position and build an environmental map, while the interface displays the detected features on a screen. Subsequently, the mapping node would publish a map and position topic, enabling the controller to determine the necessary actions to reach a specified position on the map. Additionally, an example configuration service would permit the interface to request and modify the ORB parameters.

4.3.2 ROS Node

As observed in Figure 4.4, the implementation of ROS facilitates the creation of modular systems. These systems are straightforward to construct owing to the universal protocol description, and they also permit the interchange of modules, which can be treated as functional black boxes. With this in mind, a ROS node was deployed on the Zybo Z7-20 to enable the integration of the proposed ORB accelerator with both existing and newly adapted systems.

To execute the ROS node, the CPU utilizes a tailored Linux image [70] that includes the essential ROS core utilities along with a device tree tailored to the specified architecture. This node, illustrated in Figure 4.5, is tasked with identifying the frames being transmitted to the ORB accelerator located within the FPGA (Frame Supervisor). It is also responsible for reading data from the Feature Memory (Feature Reader), publishing the identified features as a ROS topic (Topics Publisher), and responding to parameter change requests (Topics Subscriber). This combined hardware-software framework enables straightforward tuning of the ORB accelerator and facilitates its integration into broader modular systems. The hardware modules in HDMI2RGB (see Figure 4.1)require runtime configurations to initi-

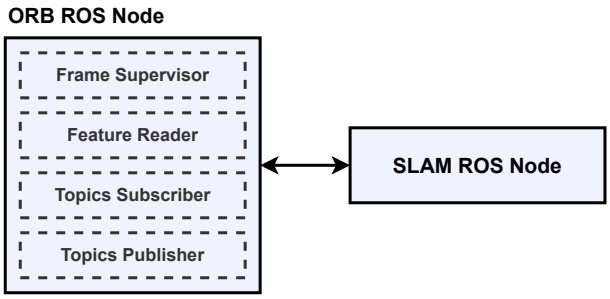


Figure 4.5: Functional blocks of the described ROS architecture.

ate the reception of an HDMI signal. These are configured to automatically detect the source resolution

amongst several other control options, and a physical memory address at which the frames can be written to is also specified should there be a need for the CPU to read the frames. To track the frame count, the node simply checks for an all-ones entry on the feature memory that signals an end of frame, and once this occurs it publishes the detected features along its descriptors to the corresponding topic. The node also subscribes to parameter change requests, which can result in a change of the FAST contrast threshold or a reset of the ORB module.

This thesis proposes a complete ORB accelerator device ready to be deployed in various application domains. To integrate it with some Simultaneous Location and Mapping (SLAM) algorithms, such as ORB-SLAM3 [71], minor modifications to their specific implementations is needed. These existing algorithms presently offer support exclusively to ROS cameras and have not been adapted to handle feature topics, which would enable the delegation of image processing to external devices.

4.4 Summary

This chapter describes the implementation of an ORB accelerator device targeted for efficient feature extraction in computer vision applications, with an emphasis on reducing latency and energy consumption. This device has been implemented on a Digilent Zybo Z7-20 board featuring an XC7Z020 SoC, which includes both a CPU and an FPGA.

The ORB accelerator assumes the critical role of processing video frames by identifying and transmitting features through Ethernet or alternate interfaces available on the platform. The ARM CPU fitted on the board runs a customised Linux image, which initialises a ROS node dedicated to configuring the accelerator and reporting detected features in a variety of applications, such as underwater mapping. In particular, the ORB accelerator was engineered with scalability in mind, enabling a configurable balance between resource usage and detection accuracy. Key adjustable parameters, including image resolution, FAST contrast threshold, and scale count, facilitate this balance by allowing performance optimisation relative to resource demands. For demonstration purposes, the device is capable of processing video inputs from an HDMI source, outputting the original frames with the detected features highlighted.

Importantly, the CPU, through the ROS node, plays a critical role in orchestrating communication between the accelerator and other components of the system, securing both data transfer and control. This seamless integration within the ROS framework affords the ORB accelerator with the adaptability needed to integrate diverse modular robotic architectures, thus promoting the efficient exchange of data among interlinked subsystems. Employing ROS in this context underscores the capacity to support the deployment of complex robotic systems, thus establishing the ORB accelerator as a multifaceted and indispensable component within advanced computer vision applications.

5

Experimental Evaluation

Contents

This chapter presents a comprehensive evaluation of the proposed Oriented FAST and Rotated BRIEF (ORB) accelerator architecture and its deployment in the Digilent Zybo Z7-20 board in a complete feature extraction system.

Accordingly, the architecture and system are first validated with standard (TUM-VI [5]) and real-world (underwater cave video, provided by the 3D-CAVE project) data-sets. Next, the accelerator is thoroughly evaluated in terms of feature extraction accuracy, scalability, hardware resource utilisation, power consumption, and energy efficiency, the chapter is concluded with a discussion on the main advantages and limitations that can still be addressed in the future.

5.1 Experimental Setup Overview

The proposed ORB accelerator was tested in two different setups with two different goals. The first intends to demonstrate reliable real-time performance and the second to prove robust construction of the ORB [31] descriptors. The next paragraphs describe these two architectures and their main features specifically designed for this evaluation.

5.1.1 Feature Overlay Demonstration

Considering the use case that motivates this thesis (real-time underwater cave mapping), it is of utmost importance to demonstrate the prototype's ability to receive a video stream and detect its features with minimal latency. To test this capability, the feature extraction system proposed in Chapter 4 was extended

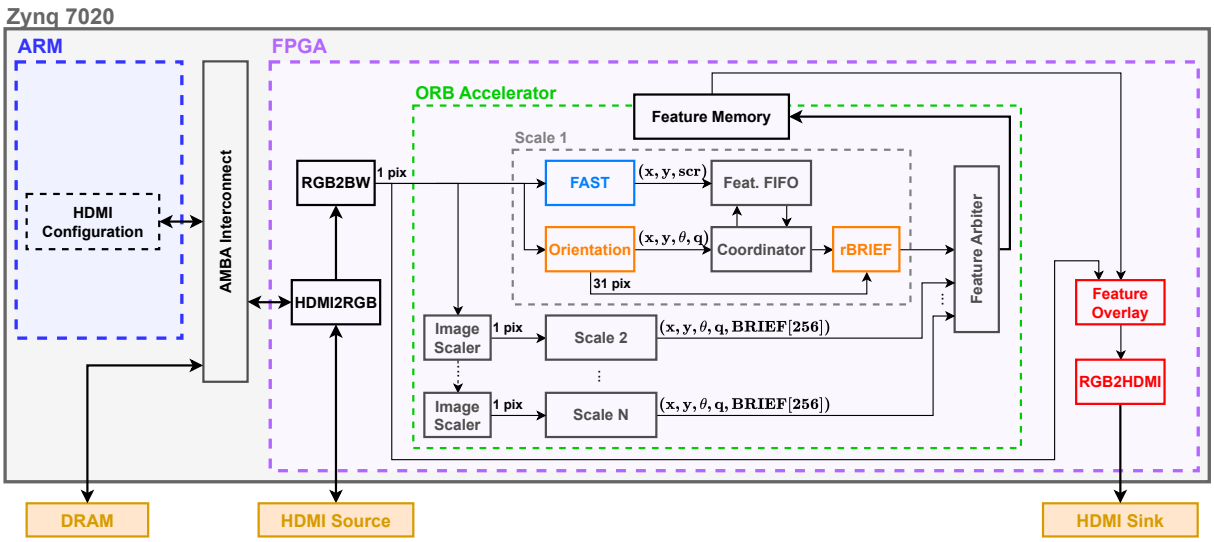


Figure 5.1: Modifications to the system architecture presented in Figure 4.1 (Chapter 4) for the real-time feature overlay validation.

to output an extracted feature overlay over the Field Programmable Gate Array (FPGA) High-Definition Multimedia Interface (HDMI) output interface (see Figure 5.1).

To validate the features extracted with the proposed accelerator, the following procedure is employed. First, a 640×480-pixel video is streamed to the HDMI receiving port at a rate of 60 Frames Per Second (fps). The ORB module is the same as previously described, but instead of reading its output from the Central Processing Unit (CPU), the feature memory is read by a dedicated Feature Overlay hardware module that overlays the detected features on the incoming frames. This module tracks the position of the pixel currently being streamed to the FPGA (using the HDMI synchronisation signals mentioned in Chapter 4) and once these coordinates coincide with the ones of a detected feature it changes that pixel's content to a distinct colour (red, green, blue or white). This means that the feature overlay is done with one frame of delay, which is not relevant for the purpose of this demonstration whose result is exemplified in Figure 5.2.

Although more comprehensive tests were performed to verify the robustness and invariance of the detected features and the constructed descriptors, it was important to implement a design that enables the detection of unexpected behaviours and poor performance of the accelerator that were otherwise hard to detect and be fixed.

Figure 5.2: Real-time extracted feature overlay demonstration. The two video streams presented side by side are the feeds being streamed to and from the Zybo Z7-20 board (left and right respectively) where the detected features are highlighted in white colour.

5.1.2 Validation

The ORB accelerator's complete output is evaluated by making use of a slightly modified version of the system described in Chapter 4. As previously stated, the ORB accelerator discards features when the descriptor constructor is busy, as such, in order to validate the Features from Accelerated and Segments

Test (FAST) [51] corner detector implementation on its own, a multiplexer (highlighted in Figure 5.3) is added that selects which features are actually written to memory (FAST features only or full ORB descriptors).

The ORB module itself was synthesised with the different configurations needed for the tests performed. The accelerator receives static images via HDMI and logs detected features into text files which are transferred to a destination computer via Secure Shell (SSH) for subsequent analysis.

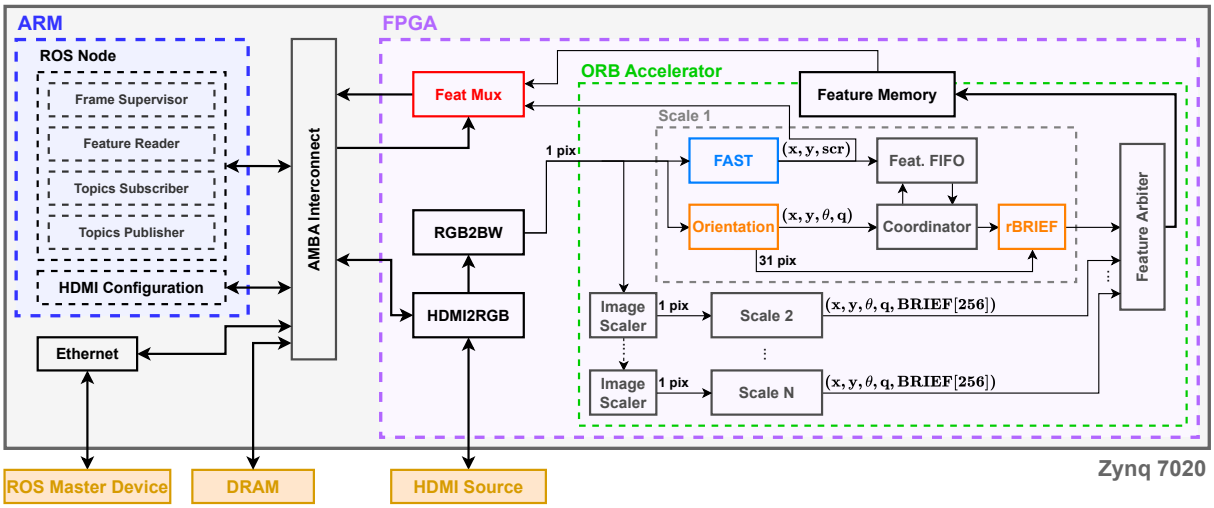


Figure 5.3: ORB accelerator device with added feature multiplexer (highlighted in red) for isolated FAST validation.

5.2 ORB Accelerator Validation

To validate and verify the feature extraction accuracy of the proposed accelerator its output is compared with baseline ORB [31] and FAST [51] algorithms implemented as part of the OpenCV [72] library.

The frames used for the considered validation experiments described below were extracted from the TUM-VI [5] data-set to ensure that the presented metrics allow for a fair comparison with the state-of-the-art implementations [4, 44, 57, 60]. Additionally, the accelerator was also tested with frames from the 3D-CAVE underwater cave data-set.

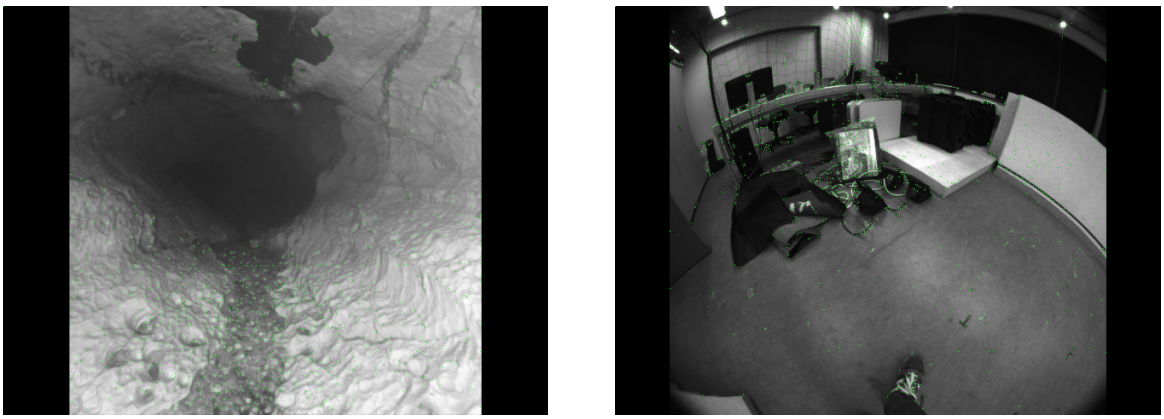
5.3 Feature Extraction Accuracy and Accelerator Scalability

In the design phase of the proposed accelerator, one key aspect was its applicability in a wide variety of scenarios, each with distinct necessities. For instance, certain implementations of the proposed ORB module might be focused on integration within embedded low-power devices. These are primarily aimed at enabling approximate position estimation and Three dimensional (3D) reconstruction with minimal

energy usage, such as in the mapping of underwater caves. In contrast, other scenarios will require highly precise feature descriptors for accurate location and mapping, where energy consumption is less of a priority, as in the case of autonomous vehicles. The ORB module has been projected to be both scalable and adaptable to these diverse requirements. Accordingly, this section provides a detailed evaluation of how modifications in the following parameters affect accuracy: FAST contrast threshold, the number of orientation sectors, and the number of scales.

5.3.1 FAST Corner Detector Validation

The hardware setup for detecting FAST corners strictly follows the original algorithm and does not incorporate any optimisations or modifications. This being said, the ORB algorithm implements a Harris corner measure [54] to rank features within an entire frame and selects the highest scoring corners. This process entails conducting complex computational tasks to calculate these corner scores and necessitates processing the entire frame to identify the top features, a strategy that conflicts significantly with the pixel-streaming method that caracheterises the accelerator's design.



(a) 3D-CAVE data-set frame.

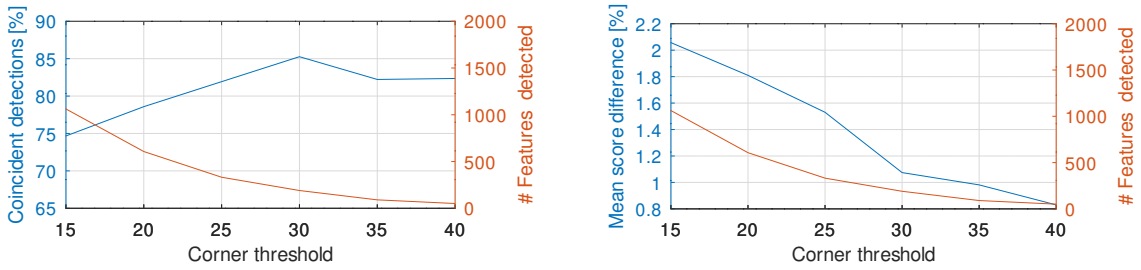
(b) TUM-VI datset frame [5].

Figure 5.4: FAST corners highlighted in green on frames processed in Zybo Z7-20 board, with a contrast threshold of 15.

The 3×3 non-maximum suppression technique employed in the hardware implementation locally eliminates features, contrary to the global ranking method employed by the OpenCV [72] implementation. Therefore, differences in feature selection are anticipated to become more pronounced at lower corner thresholds due to the increased number of detected features. Such variations depend on the frames being processed and are more pronounced when the dominant features are localised within a small area. To assess the similarity between the hardware and software outputs, two Key Performance Indicators (KPIs) were utilised: coincident detection and mean score difference (derived from the absolute sum

of the 16 pixels differences). While the first metric directly reflects the similarities between the two implementations, the mean score difference offers deeper insight into the robustness of the selected corners, which is ultimately more crucial. The tests were performed for six different contrast thresholds, for frames of the TUM-VI [5] and from the 3D-CAVE underwater cave data-set.

The images employed in this study exhibit a variety of distinctive attributes, leading to results that distinctly highlight these variations for each individual image. The initial image showcases an environment characterized by a well-distributed set of features throughout its expanse. Given this context, the findings represented in Figure 5.5(a) clearly demonstrate that the disparity in outcomes between software and hardware arises due to the differing selections of top-ranking features. As anticipated, the coincident detection and feature selection improves as the total number of detected features diminishes, which is also illustrated in Figure 5.5(a).



(a) Coincidence of detected features.

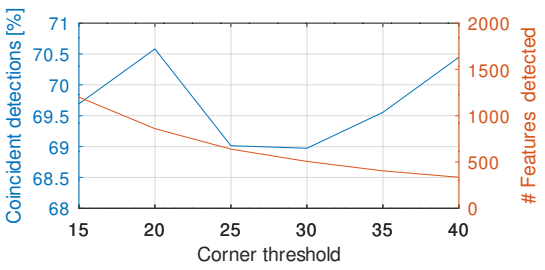
(b) Selected corners strength difference.

Figure 5.5: Comparison of Accelerator and OpenCV FAST implementations with the 3D-CAVE data-set.

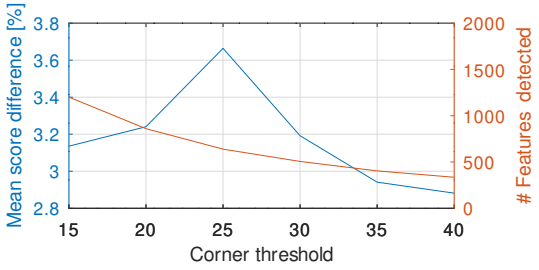
The results for the TUM-VI [5] data-set frame, however, are more variable. This image has increased complexity and feature clutter (see Figures 5.4(a) and 5.4(b)). Notably, in cluttered regions with numerous strong corners, a decrease in global feature count accentuates the impact of discarding these strong features. Therefore, for lower contrast thresholds ($[15, 25]$), changes in the number of matching detections and mean scores are not consistent, whereas at higher thresholds, the detector is sufficiently selective, making local maximum suppression effects less noticeable.

In summary, the local Non Maximum Suppression (NMS) strategy employed in the ORB accelerator visibly affects the selection of top-ranking features, as anticipated. This impact is especially pronounced in feature cluttered images and can be mitigated by raising the corner threshold to achieve consistent feature selection across the frame. Nevertheless, when focusing solely on the values and not their first derivative, the mean score differences of the selected features in software and hardware remain negligible, with a maximum variance of approximately 3.7% for the TUM-VI [5] frame (see Figure 5.5(b) and 5.6(b)).

These findings highlight robust feature detection facilities of the proposed accelerator, despite its more time-efficient selection process.



(a) Coincidence of detected features.



(b) Selected corners strength difference.

Figure 5.6: Comparison of Accelerator and OpenCV FAST implementations with the TUM-VI [5] data-set.

5.3.2 rBRIEF Descriptor Construction Validation

The process of hardware implementation for the Rotated Binary Robust Independent Elementary Features (rBRIEF) algorithm required a series of optimisations aimed at reducing the consumption of memory, multiplexing, and routing resources. By dividing the descriptor construction into 43 steps, resource utilisation was significantly reduced. However, it is imperative that this optimisation does not directly change the accelerator's outputs when compared to those produced by the software version. Another resource-related optimisation involved discretising the feature orientation into N discrete sectors. Despite this modification, the accelerator must continue to exhibit considerable rotation invariance as discussed in previously. Nevertheless, it is expected that the descriptors generated by the hardware accelerator will not match precisely with those from the software, in which the orientation has a much finer discretisation (single-precision floating-point representation). Similarly, the optimisation of the circular pattern is predicted to result in descriptors that are less robust, as it scans a reduced area, specifically, a circular area with diameter d in contrast to a $d \times d$ square area.

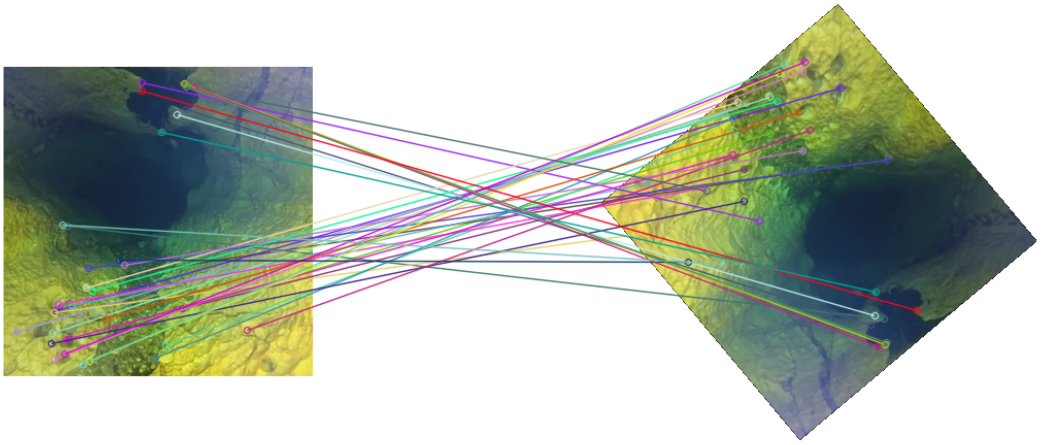


Figure 5.7: Accelerator matching after rotation.

Given that a primary objective of the rBRIEF module is the creation of rotation-invariant descriptors, the employed (KPI) is the count of positive matches for the same scene evaluated at 360 distinct orientations (between 0° and 360° with a step of 1° – see Figure 5.7 for an example of this rotation). The threshold adopted for the FAST [51] phase was established at 30, selected for producing the most analogous results to the OpenCV [72] implementation.

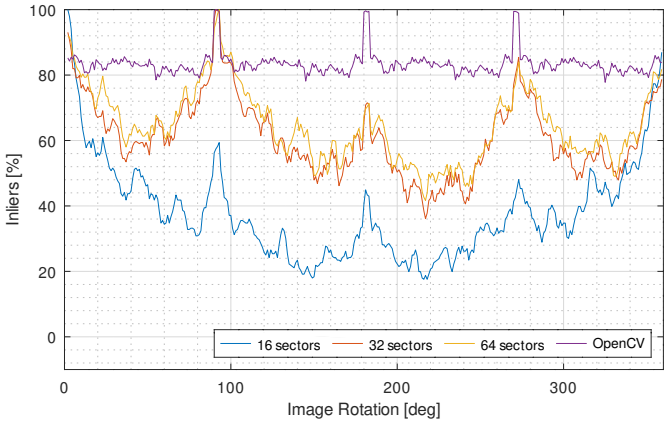


Figure 5.8: rBRIEF descriptor's robustness to rotation at 360 different angles of image rotation and for 3 different levels of orientation discretisation.

The results illustrated in Figure 5.8 exhibit a predicted enhancement in the robustness of the descriptors as the number of orientation sectors (N) increases, an improvement which is further clarified in Figure 5.9, displaying the mean percentage of correct matches over the 3 discretisation levels evaluated. It is evident that a significant improvement in accuracy, of 20%, is observed as the sector count increases from 16 to 32, whereas only a marginal gain is observed when further increasing the sectors to 64. For all three tested levels of discretisation (16, 32, and 64) a couple of main characteristics are noticeable in the progression of matching accuracy as the plane rotates: peaks in performance are noticeable at angles of 90°, 180°, 270°, and 360°; and the lowest matching percentages are noted in the second and third quadrants.

The first observation can be easily explained by considering that the rotation of the Binary Robust Independent Elementary Features (BRIEF) pattern involves trigonometric functions to rotate integer coordinates, requiring numeric approximations. At the orientations of 90°, 180°, 270°, and 360°, the sine and cosine values are either 0 or 1, thus eliminating the need for approximation, resulting in precise coordinates for the rotated BRIEF patterns. In contrast, to justify the second observation, a more thorough examination and reasoning concerning the architecture of the ORB accelerator proposed is required.

As stated before, the construction of the rBRIEF descriptor was subjected to several optimisations that resulted in an increase of its latency. While this latency is not intended to cause any direct changes to the resulting descriptor, the descriptor constructor is busy for 86 clock cycles. During this period, should more features be detected, the constructor would be unavailable for composing their descriptors

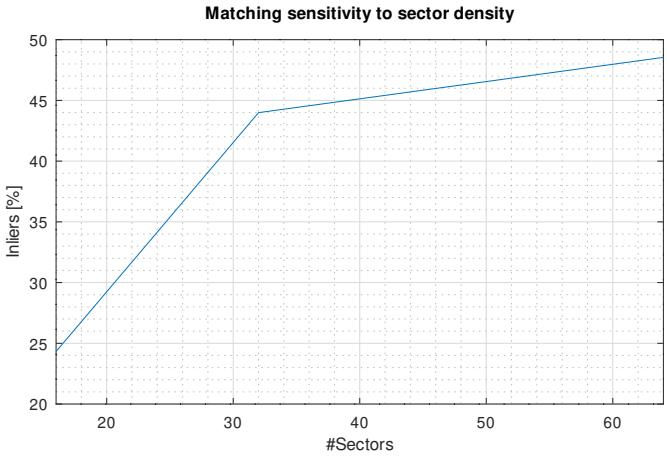
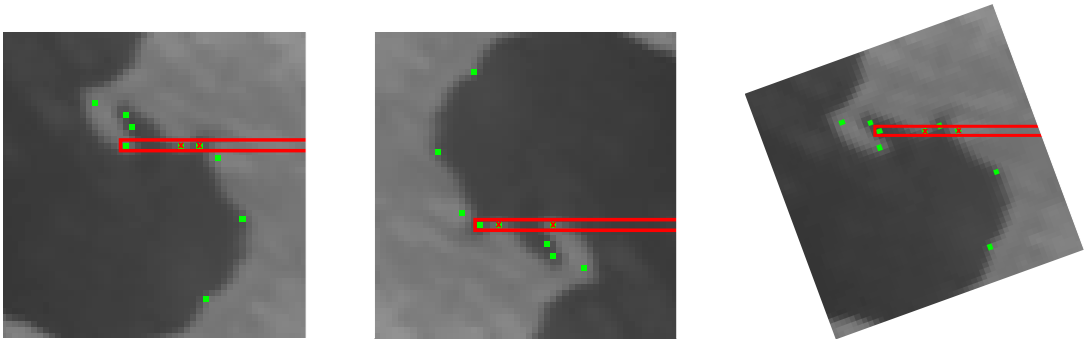


Figure 5.9: Sensitivity of the rBRIEF descriptor's robustness to rotation for 3 different levels of orientation discretisation.

and would consequently discard them. Although this constraint is not typically problematic for most real-world applications, since the image sensor moves and acquires features in various segments of the frames, in these particular tests, this characteristic of the feature extraction accelerator, together with rotation centred on the centre of the image, implies that the features discarded in the original image will not precisely coincide with those rejected in the rotated frames. This behaviour is illustrated in Figure 5.10, where a 50×50-pixel region of the 3D-CAVE underwater cave frame (Figure 5.4(a)) is rotated, clearly demonstrating that this latency obstacle results in the dismissal of different features for different rotations.



(a) Descriptor constructed at rotation 0°.

(b) Descriptor constructed at rotation 180°.

(c) Descriptor constructed at rotation 30°.

Figure 5.10: Example of discarding nearby features due to the descriptor latency. The red boxes marks the busy window and the red crosses mark the discarded features.

Despite the accuracy compromise from this discretisation, when considering popular Simultaneous Location and Mapping (SLAM) data sets [5, 73], the robustness to rotation is enough to maintain feature tracking, since the orientation of the frame does not commonly exceed an absolute value of $90°$.

5.4 ORB Accelerator Implementation Evaluation

The modifications introduced in the implementation to the ORB accelerator primarily concentrated on two fundamental efficiency metrics: the usage of hardware resources and the consumption of energy. The consideration for utilisation of hardware resources originates in the potential to execute this accelerator on cost-effective FPGA platforms. Additionally, it provides the opportunity to allocate space for the integration of other modules within the Programmable Logic (PL) fabric, which could accelerate additional tasks related to computer vision applications. The approach of utilising fewer hardware resources and opting for less costly FPGA devices overlaps with the energy efficiency objective, as these smaller devices are often characterised by reduced power consumption. Addressing energy consumption is especially crucial in portable, battery-powered embedded devices. The significance of this focus will be illustrated through a practical example in the subsequent paragraphs.

The proposed accelerator was implemented and deployed on the adopted FPGA System on Chip (SoC) (XC7Z020) with the AMD Xilinx Vivado 2020.2 toolchain. Hardware resource usage and power estimation were also obtained with the available tools for this device.

5.4.1 Hardware Resources Usage

Table 5.3 compares the hardware utilisation results obtained for the proposed ORB accelerator with state-of-the-art solutions [4, 44, 57, 60]. In most previous efforts [44, 57, 60], higher-end FPGA systems had to be used, since their architectures require substantially more hardware resources. The most similar solution is from [4], which uses a SoC of the same family, albeit with more available and used resources. This being said, the ORB accelerator here proposed manages to have a resource utilization of Look Up Tables (LUTs), Digital Signal Processors (DSPs) and Block Random Access Memorys (BRAMs) when compared to the implementation on a similar device [4], while attaining a maximum frequency that allows it to process more common 640×480-pixel frames at a frame rate of over 300 fps or 1280×720-pixel frames at over 100 fps. Furthermore, the resource utilisation for the multiple levels of orientation discretization is also presented to allow for an analysis of suitability of this accelerator for different purposes, bearing in mind the accuracy and resource usage trade-offs. The reduced hardware complexity of the proposed ORB accelerator architecture also makes it the best performing in terms of output latency, which refers to the time taken from the start of the transmission of an image to the output of a feature located at the lower right corner of that same image.

In Table 5.3 it is also possible to observe the resource usage of the proposed architecture when considering several different discretisation levels. A conclusion can be made that, since only the BRAM utilisation experiences major changes, recurring to the lower orientation sectors counts is only required when memory resources are in high demand from also other modules implemented on the FPGA. On

Table 5.1: ORB Accelerator proposed and state-of-the-art resource usage comparison.

Resources and Accuracy									
Work	**Device**	**Resolution**	**#Scales**	**Orientation**	**LUT**	**DSP**	**BRAM [Mb]**	**Latency [ms]**	**Max Freq. [MHz]**
[60]	XCZU7EV	3840x2160	1	RS-BRIEF*	62,223	668	1.62	#NP	150
[44]	XCZU3EG	640x480	4	64 sectors	76,424	80	4.32	2.5	150
[57]	Altera Stratix V	640x480	2	256 sectors	25,648	8	1.18	14.8	203
[4]	XCZ7045	640x480	1	RS-BRIEF*	56,954	111	2.81	9.1	100
This work	XC7Z020	640x480	1	16 sectors	16,179	42	0.58	3.2	100
				32 sectors	16,316		0.72		
				64 sectors	16,586		1		
			2	16 sectors	28,248	84	1.15		
				32 sectors	28,521		1.44		
				64 sectors	29,080		2		

*Uses the RS-BRIEF descriptor [4], i.e., it does not rely on the rotation of the BRIEF pattern but instead the rotation of the descriptor. # Not provided.

the other hand, the increase of the number of scales plays a much more pronounced role in increasing the memory (BRAM) and computational (LUTs and DSP) resources since it implies the replication of FAST and rBRIEF modules. Finally, the frame resolution largely impacts on the shift register utilisation, which in this platform are a subcomponent of LUT slices having made it impossible to successfully implement an accelerator with a resolution over 640×480 pixels.

5.4.2 Energy Efficiency

Energy consumption is especially relevant in the domain of portable, battery powered, embedded devices. These characteristics impose power constraints not only to maintain a suitable battery autonomy but also to keep the devices batteries weight within reasonable value. This becomes an even more pressing concern when the batteries necessary for the analysed use case of underwater cave mapping are considered.

Let us start by analysing the sensitivity of the ORB module power consumption on the weight of a future device so that the contribution of this work becomes evident. Watertight battery packs not only need to withstand the exceeding pressure of deep waters but also need to be able to deal with the internal pressure differences that result from the cell expansion under load. This results in the increase in the weight of these battery packs that can weigh 2.6 kg as in the example shown in Figure 5.11 which has a capacity of 108 Wh and a consequent energy density of 41.54 Wh / kg.

Accounting for the weight of each battery canister, a single unit is considered for this study, since a diver will have to carry such device for an extended period of time. Since the case study here present is underwater cave exploration, additional energy consumption is also accounted for by the light projectors used during dives, and a full SLAM pipeline being executed on an external device whose power consumption is referenced to [4, 44, 57, 60]. In Table 5.2 the impact of the power consumption of the ORB

Figure 5.11: Halcyon NiMH Canister battery pack used in underwater cave exploration to supply the electronic devices used such as spotlights and mapping equipment.

accelerator on the possible mapping time is presented with a variation of up to 1:33h (61%).

Table 5.2: ORB accelerator power consumption impact on battery life of a potential underwater cave mapping device.

Power Consumption				Battery Specifications			Duration		
ORB [W]	SLAM [W]	Lights [W]	Total [W]	Capacity [Ah]	Voltage [V]	Energy [Wh]	Hours [h]	Minutes [min]	Difference
0.1	5	21	26.1	9	12	108	4	8	0%
0.2			26.2				4	7	0%
0.3			26.3				4	6	-1%
0.4			26.4				4	5	-1%
2			28				3	51	-7%
3			29				3	43	-11%
4			30				3	36	-15%
5			31				3	29	-19%
6			32				3	23	-23%
10			36				3	0	-38%
15			41				2	38	-57%
16			42				2	34	-61%

Table 5.3 presents a comparison of the performance and energy efficiency between the proposed accelerator and the state-of-the-art solutions. It is possible to observe that the work here presented achieves the lowest energy consumption with energy efficiency gains between 6.7$\times$ and 31.3$\times$ over the previous works, it is the combination of the findings in Tables 5.2 and 5.3 that provides the awareness to the benefits of this thesis implementation. A mapping device equiped the designed ORB accelerator gives divers between 39 and 17 more minutes to complete their activities with gains between 18% and 6% in mapping time.

5.5 Discussion

Evaluating the ORB accelerator provided important insights into its capabilities and limitations in practical scenarios. Despite its robust performance in many cases, some trade-offs impact the real-time operation and descriptor accuracy, suggesting the need for further refinement.

Table 5.3: ORB Accelerators performance comparison table in terms of resource usage, throughput, and energy efficiency.

Accuracy							Peform. and Efficiency		
Work	**Device**	**Resolution**	**#Scales**	**Orientation**	**Latency [ms]**	**Max Freq. [MHz]**	**Power [mW]**	**Throughput [fps]**	**Energy Eff. [mJ/frame]**
[60]	XCZU7EV	3840x2160	1	RS-BRIEF*	#NP	150	5042	60	84
[44]	XCZU3EG	640x480	4	64 sectors	2.5	150	+4600	62	74
[57]	Altera Stratix V	640x480	2	256 sectors	14.8	203	4556	67	68
[4]	XCZ7045	640x480	1	RS-BRIEF*	9.1	100	1936	55.87	35
This work	XC7Z020	640x480	1	16 sectors	3.2	100	149	60	2.48
				32 sectors			159		2.65
				64 sectors			167		2.78
			2	16 sectors			290		4.8
				32 sectors			312		5.2
				64 sectors			328		5.5

*Uses the RS-BRIEF descriptor [4], i.e., it does not rely on the rotation of the BRIEF pattern but instead the rotation of the descriptor. +Energy consumption is not provided for the isolated ORB accelerator, the value was deduced based on the provided comparison against [4]. # Not provided.

Local maximum suppression

Testing the developed FAST module separated from the remaining ORB pipeline reveals that the local maximum strategy causes minor variations in selecting the top-ranking corners / features compared to the original software implementations of the ORB algorithm. While this difference can be considered insignificant for most applications, as the average score of chosen features indicates relevant selection, making the local NMS size adjustable could improve the accelerator, allowing for a more globally orientated corner selection, despite potentially missing strong and relevant corners that may be in close vicinity of each other.

Orientation discretisation

The rotation invariance analysis indicates that using a variable number of orientation sectors enables the ORB module to be adapted for various platforms, striking a balance between accuracy and resource efficiency. However, the blockage of the constructor during 86 clock cycles proved to be a bottleneck in accurately matching features in rotated images, with no significant improvements observed from increasing discretisation from 32 to 64 or more orientation sectors. Although the current performance is adequate for most mapping applications, this suggests that further design modifications could enhance the accelerator's robustness for applications like object identification, which has a greater reliance on rotation invariance.

Scalability

The parametrization of this feature extraction module that allows it to be dimensioned according to the needs of the systems to which it is deployed has proved to make it scalable to specific platforms and

adaptable to specific accuracy requirements. This being said, it was not possible to test its integration in larger platforms thus the conclusions to its full potential are limited to mid-resolution images and low scale invariance because it was not possible to implement an accelerator configured to meet higher requisites that would result in a resource utilisation compatible with available device. Although this thesis targeted a low-cost FPGA platform, the continuation of this work should target larger platforms to permit a more complete analysis of this feature.

5.6 Summary

This study evaluates the efficacy and adaptability of an ORB accelerator engineered for real-time, efficient feature detection and descriptor construction. The experimental setup involved both live video streams and static images to validate the ORB accelerator's capacity for reliable feature detection and robust descriptor building. The results indicated consistent and efficient real-time operations with minimal latency, essential for tasks like underwater cave mapping, where real-time feature overlay provides immediate visual feedback. The hardware effectively identifies key features in both static and dynamic scenes, demonstrating the module's robustness.

Furthermore, the ORB accelerator's efficiency was verified through comparative tests against standard software methods, using renowned data-sets for in-depth benchmarking. Performance metrics showed that discrepancies in hardware versus software corner selection were minor and manageable by adjusting parameters like contrast threshold and orientation discretization. For example, higher contrast thresholds improved feature selection and matching consistency in cluttered environments. Evaluations of rotational invariance indicated that efforts to minimise resource use, such as orientation sector discretization, led to significant improvements without compromising descriptor accuracy. Overall, the accelerator's design achieves a notable balance between resource efficiency and robustness, making it suitable for embedded systems with power and space limitations.

The reduced resource usage that characterises this accelerator allows for its integration into low-power devices, improving their operational duration, which is a considerable advantage in remote and resource-constrained settings. Although this evaluation was limited to mid-resolution image testing, the findings suggest the potential for future adaptation to larger platforms, where increased processing demands and higher resolutions could further confirm the scalability and adaptability of this accelerator design.

6

Conclusion

Contents

This thesis proposes an Oriented FAST and Rotated BRIEF (ORB) hardware accelerator that advances efficient feature extraction in real-time embedded systems, specifically targeting low-power and resource-limited applications like underwater cave mapping. It meets crucial objectives in power efficiency and processing speed. Tests on both static and dynamic images confirmed its capability for swift, accurate feature extraction with minimal delay. Embedded in an Field Programmable Gate Array (FPGA)-based System on Chip (SoC), the accelerator demonstrated excellent scalability and adaptability, catering to various embedded computer vision applications.

While the system achieves successful results, it is nonetheless subject to some constraints attributed to its inherent design and the particular platform selected for testing purposes. Specifically, the system was restricted to working with mid-resolution images, a limitation imposed by the available hardware. Additionally, introduced trade-offs impact its ability to maintain robust rotation invariance, a characteristic of significant importance for applications necessitating increased orientation adaptability. Addressing these constraints presents a path for future development. This includes efforts to expand scalability to accommodate higher-resolution images, refine its rotation invariance capabilities, and enhance the system's overall configurability. Such advancements have the potential to significantly extend the accelerator's range of applicability across diverse fields and substantially boost its performance, particularly within complex computer vision systems.

6.1 Conclusions

This thesis details the design, implementation, and evaluation of an efficient ORB hardware accelerator designed for embedded systems that require low power and real-time functionality. Driven by the necessity for efficient, portable mapping tools in challenging contexts like underwater cave exploration, this work tackled major computational and energy limitations of current visual mapping methods. The central goal was to design a feature extraction accelerator that functions effectively under highly restrictive resource constraints while delivering reliable low-latency feature extraction performance across diverse application domains.

By integrating novel design strategies and specific optimizations, the ORB accelerator reached real-time performance while using minimal power. The design features an optimised Features from Accelerated and Segments Test (FAST) feature detector and a dedicated Rotated Binary Robust Independent Elementary Features (rBRIEF) constructor, enabling quick feature extraction with maintained accuracy. Experiments revealed that the accelerator effectively detects and processes essential features, preserving orientation and scale invariance with minimal computational cost. Consequently, the device significantly improves energy efficiency and reduces latency over traditional software-based and previous FPGA-based approaches, crucial for battery-reliant and resource-limited applications.

The findings highlight the importance of the ORB accelerator for real-time embedded use, like underwater cave mapping, where quick feature extraction is critical for immediate feedback to divers. Delivering precise feature data in real time minimizes the requirement for post-dive data analysis, speeding up exploration. Additionally, its modular design allows adaptation to other domains dependent on fast image-based localisation and mapping, such as robotics, autonomous navigation, and environmental monitoring.

This thesis not only tackles the primary aim of underwater mapping but also underscores the expansive potential of specialised hardware for feature extraction. The ORB accelerator's successful implementation as an FPGA-based system compatible with ROS showcases the capability for deeper integration into complete Simultaneous Location and Mapping (SLAM) pipelines or other computer vision tasks that require rapid real-time feature extraction. The work's contributions establish a foundation for future uses and modifications of the ORB accelerator, ensuring its continued versatility and impact in embedded computer vision.

This thesis concludes that an ORB hardware accelerator, specifically optimised for energy efficiency and computational speed, effectively supports real-time embedded feature extraction in complex environments. The findings confirm the accelerator's potential to enhance image-based mapping and localisation, significantly contributing to underwater cave mapping and various other industries reliant on efficient visual data processing.

6.2 Future Work Guidelines

The ORB accelerator presented revealed limitations that can be further mitigated through the implementation of further optimisations or specific design changes. As is, the module is constrained in its ability to: pick top scoring features with a global frame perspective; describe orientation due to the discretisation employed; and produce descriptors for all the features selected by the non-maximum suppression stage.

Feature selection

The strategy used to select the highest scoring features operates in a sliding window, making this method unaware of the context of the global frame. The findings in Chapter 5 reveal that while the selected features still perform well in comparison the ORB implementation in software when looking at mean score of the selected features, it is visible that the actual features selected differ in up to 25%. To maintain the pixel streaming architecture that is the basis of this design, two alternatives can be highlighted.

One option to keep the selection at the output of the FAST module would be to make the size of the Non Maximum Suppression (NMS) window parametrizable such that larger windows could be tested, but this is not guaranteed to provide a better selection of features nor does it completely mitigate the locality

issue being addressed. Solutions that allow the accelerator to have a global perspective on the feature ranking will most likely require that all detected features have their descriptors constructed and once the entire frame has been processed the features can be ordered by their scores using dedicate sorting hardware such as [74]. While this would imply the increase of resource usage for the construction of a number of descriptors that could be up to 9 times larger (3×3), it would mean a selection according to the specifications of the original ORB algorithm.

Orientation discretisation

The suggested discretisation of the feature orientation in N discrete sectors is reasonable, especially when considering that the comparison coordinates are integer values that inevitably require approximations. This being said, the method described in this thesis requires the pre computation of N Binary Robust Independent Elementary Features (BRIEF) patterns that greatly reduces the computations done by the accelerator, but increases the use of memory resources that are used to store these patterns. While this limitation did not prevent the implementation of a robust feature extraction accelerator on the targeted device, a proposition has been made in [4] that could mitigate memory usage. Mateusz Wasala et al. [4] propose a different comparison pattern named RS-BRIEF that is composed of consecutively rotated coordinate pairs, meaning that the rotation of the pattern results in a simple shift of the descriptor bits which greatly simplifies the reorientation process and requires a single pattern to be stored. However, this work recognises that this technique is likely to produce descriptors that do not perform as well as the rBRIEF regarding orientation and matching. The fact that they also fail to provide specific metrics for rotation invariance deemed it not viable for the context of this thesis but it can be something further explored in the future.

Descriptor construction

The descriptor composition stage was identified as the primary performance bottleneck of the described ORB accelerator. While it successfully builds descriptors that facilitate robust feature matching, even under challenging test conditions, its delay and rejection of features while it processes others lead to poorer matching of identical rotated images. Addressing this limitation should be a key focus for future developments.

Two strategies could be considered: replicating descriptor constructors, which would greatly increase memory usage since pattern memories must also be replicated; or buffering input image lines for features that would otherwise be ignored while the constructor is busy. This latter approach, considering the FPGA implementation, would only slightly increase resource usage as Block Random Access Memorys (BRAMs) are already employed for window buffers, storing only 31×31 pixels. Allocating one

memory port for writing and the other for reading allows continuous input buffering during feature descriptor construction. Although this requires a major design overhaul, it aligns with the pixel streaming strategy and could significantly enhance the accelerator's performance.

6.3 Final Remarks

This thesis illustrates the potential of a specialised hardware solution for efficient low-power feature extraction. The ORB accelerator here presented significantly advances embedded computer vision by overcoming the constraints of limited resources and making it easily integrable in existing systems with its Robot Operating System (ROS) compatibility. The proposed future improvements could broaden its use, enhancing its versatility in areas such as robotics, autonomous navigation, and environmental monitoring. This work lays the groundwork for developing more sophisticated computer vision systems that facilitate novel applications in demanding environments.

Bibliography

[1] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[2] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*. Springer, 2006, pp. 404–417.

[3] Zynq 7000s block diagram. Retrieved January 8, 2024. [Online]. Available: https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html

[4] R. Liu, J. Yang, Y. Chen, and W. Zhao, "eslam: An energy-efficient accelerator for real-time orb-slam on fpga platform," in *Proceedings of the 56th Annual Design Automation Conference 2019*, ser. DAC '19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3316781.3317820

[5] D. Schubert, T. Goll, N. Demmel, V. Usenko, J. Stueckler, and D. Cremers, "The tum vi benchmark for evaluating visual-inertial odometry," in *International Conference on Intelligent Robots and Systems (IROS)*, October 2018.

[6] J. K. Patil and R. Kumar, "Analysis of content based image retrieval for plant leaf diseases using color, shape and texture features," *Engineering in agriculture, environment and food*, vol. 10, no. 2, pp. 69–78, 2017.

[7] X. E. Pantazi, D. Moshou, A. A. Tamouridou, and S. Kasderidis, "Leaf disease recognition in vine plants based on local binary patterns and one class support vector machines," in *Artificial Intelligence Applications and Innovations: 12th IFIP WG 12.5 International Conference and Workshops, AIAI 2016, Thessaloniki, Greece, September 16-18, 2016, Proceedings 12*. Springer, 2016, pp. 319–327.

[8] K. A. Khan, P. Shanir, Y. U. Khan, and O. Farooq, "A hybrid local binary pattern and wavelets based approach for eeg classification for diagnosing epilepsy," *Expert Systems with Applications*, vol. 140, p. 112895, 2020.

[9] D. Nistér, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*, vol. 1. Ieee, 2004, pp. I–I.

[10] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "Monoslam: Real-time single camera slam," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[11] J. Engel, T. Schöps, and D. Cremers, "Lsd-slam: Large-scale direct monocular slam," in *European conference on computer vision*. Springer, 2014, pp. 834–849.

[12] R. A. Newcombe, S. J. Lovegrove, and A. J. Davison, "Dtam: Dense tracking and mapping in real-time," in *2011 international conference on computer vision*. IEEE, 2011, pp. 2320–2327.

[13] R. Mur-Artal, J. M. M. Montiel, and J. D. Tardos, "Orb-slam: a versatile and accurate monocular slam system," *IEEE transactions on robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[14] J. Nikolic, J. Rehder, M. Burri, P. Gohl, S. Leutenegger, P. T. Furgale, and R. Siegwart, "A synchronized visual-inertial sensor system with fpga pre-processing for accurate real-time slam," in *2014 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2014, pp. 431–437.

[15] J. K. Makhubela, T. Zuva, and O. Y. Agunbiade, "A review on vision simultaneous localization and mapping (vslam)," in *2018 International Conference on Intelligent and Innovative Computing Applications (ICONIC)*, 2018, pp. 1–5.

[16] B. L. MacDonald, J. C. Chatters, E. G. Reinhardt, F. Devos, S. Meacham, D. Rissolo, B. Rock, C. Le Maillot, D. Stalla, M. D. Marino *et al.*, "Paleoindian ochre mines in the submerged caves of the yucatán peninsula, quintana roo, mexico," *Science advances*, vol. 6, no. 27, p. eaba1219, 2020.

[17] Z. Lv, P. Zhang, and J. Atli Benediktsson, "Automatic object-oriented, spectral-spatial feature extraction driven by tobler's first law of geography for very high resolution aerial imagery classification," *Remote Sensing*, vol. 9, no. 3, p. 285, 2017.

[18] J. Huo, C. Zhou, B. Yuan, Q. Yang, and L. Wang, "Real-time dense reconstruction with binocular endoscopy based on stereonet and orb-slam," *Sensors*, vol. 23, no. 4, p. 2074, 2023.

[19] P. J. Stooke, "The geology of amalthea," *Earth, Moon, and Planets*, vol. 64, no. 2, pp. 187–197, 1994.

[20] M. Bagheri-Gavkosh, S. M. Hosseini, B. Ataie-Ashtiani, Y. Sohani, H. Ebrahimian, F. Morovat, and S. Ashrafi, "Land subsidence: A global challenge," *Science of The Total Environment*, vol. 778, p. 146193, 2021.

[21] D. Ford and P. D. Williams, *Karst hydrogeology and geomorphology*. John Wiley & Sons, 2007.

[22] G. Balázs, J. Vörös, B. Lewarne, and G. Herczeg, "A new non-invasive in situ underwater dna sampling method for estimating genetic diversity," *Evolutionary Ecology*, vol. 34, no. 4, pp. 633–644, 2020.

[23] P. Kambesis, "The importance of cave exploration to scientific research," *Journal of cave and karst studies*, vol. 69, no. 1, pp. 46–58, 2007.

[24] S. Rahman, A. Q. Li, and I. Rekleitis, "Svin2: An underwater slam system using sonar, visual, inertial, and depth sensor," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2019, pp. 1861–1868.

[25] P. Hambarde, S. Murala, and A. Dhall, "Uw-gan: Single-image depth estimation and image enhancement for underwater images," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–12, 2021.

[26] T. M. Iliffe and C. Bowen, "Scientific cave diving," *Marine Technology Society Journal*, vol. 35, no. 2, pp. 36–41, 2001.

[27] N. Weidner, S. Rahman, A. Q. Li, and I. Rekleitis, "Underwater cave mapping using stereo vision," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5709–5715.

[28] D. Giordan, D. Godone, M. Baldo, M. Piras, N. Grasso, and R. Zerbetto, "Survey solutions for 3d acquisition and representation of artificial and natural caves," *Applied Sciences*, vol. 11, no. 14, p. 6482, 2021.

[29] G. Zhang, B. Shang, Y. Chen, and H. Moyes, "Smartcavedrone: 3d cave mapping using uavs as robotic co-archaeologists," in *2017 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2017, pp. 1052–1057.

[30] D. Castells-Rufas, V. Ngo, J. Borrego-Carazo, M. Codina, C. Sanchez, D. Gil, and J. Carrabina, "A survey of fpga-based vision systems for autonomous cars," *IEEE Access*, vol. 10, pp. 132 525–132 563, 2022.

[31] E. Rublee, V. Rabaud, K. Konolige, and G. Bradski, "Orb: An efficient alternative to sift or surf," in *2011 International conference on computer vision*. Ieee, 2011, pp. 2564–2571.

[32] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[33] R. Kulkarni, S. Kulkarni, S. Dabhane, N. Lele, and R. Paswan, “An automated computer vision based system for bottle cap fitting inspection,” in *2019 Twelfth International Conference on Contemporary Computing (IC3)*. IEEE, 2019, pp. 1–5.

[34] G. Bresson, Z. Alsayed, L. Yu, and S. Glaser, “Simultaneous localization and mapping: A survey of current trends in autonomous driving,” *IEEE Transactions on Intelligent Vehicles*, vol. 2, no. 3, pp. 194–220, 2017.

[35] C. R. McBryde and E. G. Lightsey, “A star tracker design for cubesats,” in *2012 IEEE Aerospace Conference*. IEEE, 2012, pp. 1–14.

[36] J. Gong, J. Gong, and J. Tian, “Using star trackers for space surveillance.”

[37] S. Oh, J.-H. You, and Y.-K. Kim, “Fpga acceleration of bolt inspection algorithm for a high-speed embedded machine vision system (iccas 2019),” in *2019 19th International Conference on Control, Automation and Systems (ICCAS)*. IEEE, 2019, pp. 1070–1073.

[38] S.-C. Lin, Y. Zhang, C.-H. Hsu, M. Skach, M. E. Haque, L. Tang, and J. Mars, “The architectural implications of autonomous driving: Constraints and acceleration,” in *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, 2018, pp. 751–766.

[39] J. Jiang, G. J. Zhang, X. Wei, and X. Li, “Rapid star tracking algorithm for star sensor,” *IEEE Aerospace and Electronic Systems Magazine*, vol. 24, no. 9, pp. 23–33, 2009.

[40] P. E. DO CT OR OF, “Machine perception of three-dimensional, so lids,” Ph.D. dissertation, MASSACHUSETTS INSTITUTE OF TECHNOLOGY, 1961.

[41] J. Iglhaut, C. Cabo, S. Puliti, L. Piermattei, J. O’Connor, and J. Rosette, “Structure from motion photogrammetry in forestry: A review,” *Current Forestry Reports*, vol. 5, pp. 155–168, 2019.

[42] J. L. Schonberger and J.-M. Frahm, “Structure-from-motion revisited,” in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 4104–4113.

[43] M. Servières, V. Renaudin, A. Dupuis, and N. Antigny, “Visual and visual-inertial slam: State of the art, classification, and experimental benchmarking,” *Journal of Sensors*, vol. 2021, pp. 1–26, 2021.

[44] V. Vemulapati and D. Chen, “Fslam: an efficient and accurate slam accelerator on soc fpgas,” in *2022 International Conference on Field-Programmable Technology (ICFPT)*, 2022, pp. 1–9.

[45] F. Huang, H. Yang, X. Tan, S. Peng, J. Tao, and S. Peng, “Fast reconstruction of 3d point cloud model using visual slam on embedded uav development platform,” *Remote Sensing*, vol. 12, no. 20, p. 3308, 2020.

[46] J. Li, G. Deng, W. Zhang, C. Zhang, F. Wang, and Y. Liu, “Realization of cuda-based real-time multi-camera visual slam in embedded systems,” *Journal of Real-Time Image Processing*, vol. 17, pp. 713–727, 2020.

[47] K. R. Beevers and W. H. Huang, “An embedded implementation of slam with sparse sensing,” in *2008 IEEE International Conference on Robotics & Automation (ICRA 2008)*, 2007, pp. 1–6.

[48] C. Friess, V. Niculescu, T. Polonelli, M. Magno, and L. Benini, “Fully onboard slam for distributed mapping with a swarm of nano-drones,” *IEEE Internet of Things Journal*, 2024.

[49] M. R. Gkeka, A. Patras, N. Tavoularis, S. Piperakis, E. Hourdakis, P. Trahanias, C. D. Antonopoulos, S. Lalis, and N. Bellas, “Reconfigurable system-on-chip architectures for robust visual slam on humanoid robots,” *ACM Transactions on Embedded Computing Systems*, vol. 22, no. 2, pp. 1–29, 2023.

[50] M. Bloesch, S. Omari, M. Hutter, and R. Siegwart, “Robust visual inertial odometry using a direct ekf-based approach,” in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 298–304.

[51] E. Rosten and T. Drummond, “Fusing points and lines for high performance tracking,” in *Tenth IEEE International Conference on Computer Vision (ICCV’05) Volume 1*, vol. 2, 2005, pp. 1508–1515 Vol. 2.

[52] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, “Brief: Binary robust independent elementary features,” in *Computer Vision–ECCV 2010: 11th European Conference on Computer Vision, Heraklion, Crete, Greece, September 5-11, 2010, Proceedings, Part IV 11*. Springer, 2010, pp. 778–792.

[53] E. Rosten and T. Drummond, “Machine learning for high-speed corner detection,” in *Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006. Proceedings, Part I 9*. Springer, 2006, pp. 430–443.

[54] C. Harris, M. Stephens *et al.*, “A combined corner and edge detector,” in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244.

[55] P. L. Rosin, “Measuring corner properties,” *Computer Vision and Image Understanding*, vol. 73, no. 2, pp. 291–307, 1999.

[56] J. S. Beis and D. G. Lowe, “Shape indexing using approximate nearest-neighbour search in high-dimensional spaces,” in *Proceedings of IEEE computer society conference on computer vision and pattern recognition*. IEEE, 1997, pp. 1000–1006.

[57] W. Fang, Y. Zhang, B. Yu, and S. Liu, “Fpga-based orb feature extraction for real-time visual slam,” in *2017 International Conference on Field Programmable Technology (ICFPT)*. IEEE, 2017, pp. 275–278.

[58] J. Weberruss, L. Kleeman, D. Boland, and T. Drummond, “Fpga acceleration of multilevel orb feature extraction for computer vision,” in *2017 27th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2017, pp. 1–8.

[59] R. Eyvazpour, M. Shoaran, and G. Karimian, “Hardware implementation of slam algorithms: a survey on implementation approaches and platforms,” *Artificial Intelligence Review*, vol. 56, no. 7, pp. 6187–6239, 2023.

[60] M. Wasala, H. Szolc, and T. Kryjak, “An efficient real-time fpga-based orb feature extraction for an uhd video stream for embedded visual slam,” *Electronics*, vol. 11, no. 14, p. 2259, 2022.

[61] T. Peng, D. Zhang, D. L. N. Hettiarachchi, and J. Loomis, “An evaluation of embedded gpu systems for visual slam algorithms,” *Electronic Imaging*, vol. 32, pp. 1–6, 2020.

[62] M. Li, K. Arning, L. Vervier, M. Ziefle, and L. Kobbelt, “Influence of temporal delay and display update rate in an augmented reality application scenario,” in *Proceedings of the 14th International Conference on Mobile and Ubiquitous Multimedia*, 2015, pp. 278–286.

[63] S. Se, H.-K. Ng, P. Jasiobedzki, and T.-J. Moyung, “Vision based modeling and localization for planetary exploration rovers,” in *55th International Astronautical Congress of the International Astronautical Federation, the International Academy of Astronautics, and the International Institute of Space Law*, 2004, pp. U–2.

[64] R. B. Miller, “Response time in man-computer conversational transactions,” in *Proceedings of the December 9-11, 1968, fall joint computer conference, part I*, 1968, pp. 267–277.

[65] Spiral Multiplier Block Generator, “Spiral multiplier block generator.” [Online]. Available: https://spiral.ece.cmu.edu/mcm/gen.html

[66] A. P. Tafti, A. Baghaie, A. B. Kirkpatrick, J. D. Holz, H. A. Owen, R. M. D’Souza, and Z. Yu, “A comparative study on the application of sift, surf, brief and orb for 3d surface reconstruction of electron microscopy images,” *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, vol. 6, no. 1, pp. 17–30, 2018.

[67] J. Zhu, H. Li, and T. Zhang, “Camera, lidar, and imu based multi-sensor fusion slam: A survey,” *Tsinghua Science and Technology*, vol. 29, no. 2, pp. 415–429, 2024.

[68] J. Wang and D. Herath, “What makes robots? sensors, actuators, and algorithms,” in *Foundations of Robotics: A Multidisciplinary Approach with Python and ROS*. Springer, 2022, pp. 177–203.

[69] J. Kramer and M. Scheutz, “Development environments for autonomous mobile robots: A survey,” *Autonomous Robots*, vol. 22, pp. 101–132, 2007.

[70] Xilinx Inc., “PetaLinux Tools,” 2018, version 2018.2, [Online]. Available: urlhttps://github.com/Xilinx/PetaLinux.

[71] C. Campos, R. Elvira, J. J. G. Rodríguez, J. M. Montiel, and J. D. Tardós, “Orb-slam3: An accurate open-source library for visual, visual–inertial, and multimap slam,” *IEEE Transactions on Robotics*, vol. 37, no. 6, pp. 1874–1890, 2021.

[72] OpenCV, “Opencv - open source computer vision library.” [Online]. Available: https://opencv.org/

[73] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, “The euroc micro aerial vehicle datasets,” *The International Journal of Robotics Research*, 2016. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033

[74] N. Matsumoto, K. Nakano, and Y. Ito, “Optimal parallel hardware k-sorter and top k-sorter, with fpga implementations,” in *2015 14th International Symposium on Parallel and Distributed Computing*, 2015, pp. 138–147.