Extracting Distinctive Features from Light-Field Images

José Maria Abecasis Teixeira

Thesis to obtain the Master of Science Degree in

Electrical and Computer Engineering

Supervisors

Prof. João Miguel Duarte Ascenso

Prof. Fernando Manuel Bernardo Pereira

Prof. Catarina Isabel Carvalheiro Brites

Examination Committee

Chairperson: Prof. José Eduardo Charters Ribeiro da Cunha Sanguino

Supervisor: Prof. João Miguel Duarte Ascenso

Members of the Committee: Prof. Luís Filipe Barbosa de Almeida Alexandre

May 2017

"*The primary cause of unhappiness is never the situation but your thoughts about it"*

-Eckhart Tolle

Acknowledgements

This Thesis was funded by Fundação para a Ciência e Tecnologia through the BIC (Bolsa de Iniciação Científica) Research Grant.

First and foremost I would like to thank my supervisors from IST, Professor João Ascenso, Professor Catarina Brites, and Professor Fernando Pereira, for their unconditional support throughout both the research, and development phase of this thesis. Without their priceless advice and contributions, this thesis would not have been possible.

I would like to thank my parents, for all the support throughout my high-school and college years. The principles and values which both of them passed on to me allowed me to always push myself to the best of my abilities.

I would also like to thank my friends, for the emotional support throughout the (occasionally stressful) time spent working on this thesis. A special thank you to my friends at IST: João Franco, João Rocha e Melo, Hugo da Silva, Bernardo Almeida, Manel Ávila de Melo, Bernardo Marques, Manel Ribeiro, João Galamba, Manel Beja da Costa, Nuno Sousa, and Miguel Monteiro. Who accompanied me throughout my years at IST, and with whom I developed a very deep sense of friendship, and comradery.

As a final acknowledgment, I would like to thank Ederzito Antonio Macedo Lopes, for smashing Portugal on to the front of glory at the 2016 European Football Cup. Thank you Eder.

Resumo

As tecnologias de representação visual têm estado em constante desenvolvimento desde o aparecimento da fotografia a preto e branco. Esta área de conhecimento tem estado sujeita a uma constante mudança sempre com o objetivo de obter representações mais detalhadas e realísticas do mundo visual. O desenvolvimento nesta área levou ao aparecimento de famosos formatos de representação visual tais como fotografia a cores e vídeo, bem como de outros formatos mais recentes e ricos em informação tais como *point-clouds*, hologramas e *light-fields.* Estes últimos são obtidos de diversas formas, uma das quais as chamadas *lenslet light-field cameras* (e.g. Lytro), e são uma representação do mundo visual extremamente rica, uma vez que através de um único light-field é possível capturar informação relativa a várias direções da luz, obtendo assim informação acerca da profundidade da cena, e permitindo uma representação de vários planos de foco.

Uma vez que o formato *light-field* popularizou-se recentemente, pouca pesquisa foi feita com o objetivo de o processar, transmitir, manipular, ou mesmo realizar uma extração semântica da informação que este contém. Esta Tese de mestrado tem como objetivo o desenvolvimento de uma solução de extração de características em imagens *light-field,* estas características são uma redução da informação contida dentro de uma imagem de modo a representa-la de uma maneira eficiente, compacta, discriminativa. Primeiro, é realizada uma revisão das atuais soluções de extração de características mais relevantes, incluindo técnicas de extração de características concebidas para formatos mais usuais como fotografia 2D, bem como as poucas técnicas desenvolvidas para formatos mais recentes e ricos em informação, tais como *light-fields*. De seguida, novos métodos de comparação de *light-fields*, e uma nova base de dados de imagens *light-field*, apropriada para reconhecimento de imagem que foi essencial para o desenvolvimento deste trabalho são apresentados, a base de dados contem uma série de transformações de imagem de modo a testar a robustez de descritores de imagem (técnicas de extração de características), enquanto os métodos propostos utilizam descritores de imagem 2D de uma maneira alternativa para explorar a informação extra contida num *light-field* de modo a melhorar consideravelmente o processo de comparação de *light-fields*. No contexto desta tese foram também propostos novos métodos de deteção e extração de características adaptados a *light-fields,* estes métodos utilizam uma reparametrização do *light-field*, chamada *Epipolar Plane Image* ou EPI para explorar de uma maneira eficaz a informação contida num light-field e melhorar em relação a métodos de deteção e extração de características para formatos 2D já existentes.

Palavras-chave: Light-fields, função plenoptica, extração de características, detetores, descritores.

Abstract

Visual representation technology has been constantly evolving since the appearance of black and white photography. The intensive research in this area has led to significant advances with the aim of achieving a more accurate and realistic representations of the visual world. Some examples are the colour photography and video visual representation formats, as well as more recent formats such as point-clouds, holograms, and light-fields. Light-fields can be acquired by the so-called light-field camera (e.g. Lytro) and is a rather rich representation of the visual world, since it allows the capture of information of every angle of incidence of every light ray in the scene in a single photograph, thus obtaining information about the depth of the objects of the visual scene, and allows to obtain a 2D image from every focus plane.

Due to the recent appearance of light-fields, not much research has been made in light-field manipulation, coding and transmission or even the extraction of semantic information. This M.Sc. thesis has the objective of proposing novel feature detection and extraction methods, specifically tailored for light-field images, features are, in a nutshell a reduction of the visual information contained in an image in order to represent it in an efficient, compact, and discriminative way. First, a review of the most relevant feature detection and extraction solutions for a variety of visual representations, such as 2D images, as well as more recent and richer visual formats such as light-fields is made. Next, novel matching methods adapted to light-fields, as well as a light-field image retrieval dataset, which was essential for the development of this Thesis are presented, this database contains a series of image transformations used to test the robustness of image descriptors (techniques of feature extraction), while the proposed methods use 2D image descriptors in an alternate way in order to considerably improve the process of light-field matching. In the context of this Thesis, novel methods for feature detection and feature extraction for light-field images are also proposed, these methods use a parametrization of the light-field, called Epipolar Plane Image, or EPI, to exploit in an effective way the visual information contained in a light-field, and improve on already existing feature detection and extraction methods for 2D image formats.

Key-words: Light-fields, plenoptic function, feature extraction, detectors, descriptors.

Table of Contents

List of Figures

List of Tables

Acronyms

2D	Two Dimensional
3D	Three Dimensional
4D	Four Dimensional
5D	Five Dimensional
AGAST	Adaptive and Generic Accelerated Segmented Test
APM	Accumulative Perspective Matching
BFM	Brute Force Matcher
BoV	Bag of Visual Words
BRAND	Binary Robust Appearance and Normal Descriptor
BRIEF	Binary Robust Independent Elementary Features
BRISK	Binary Robust Invariant Scalable Key-Points
CenSurE	Center Surround Extrema
CPU	Central Processing Unit
CVM	Central View Matching
DenseSIFT	Dense Scale-Invariant Feature Transform
DoG	Difference of Gaussians
EPFL	Ecole Polytechnique Fédérale de Laussane
EPI	Epipolar Plane Image
ETH	Eidgenössische Technische Hochschule
FAST	Fast Accelerated Segmented Test
FPR	False Positive Rate
FREAK	Fast Retina Key-Point
GB	Gigabyte
GLOH	Gradient Location and Orientation Histogram

HOG	Histogram of Oriented Gradients
LF	Light-Field
LIDAR	Light and Radar
LISAD	Light Field Scale and Depth
LoG	Laplacian of Gaussians
MAP	Mean Average Precision
MSER	Maximally Stable Extremal Regions
OpenCV	Open Computer Vision
ORB	Oriented FAST Rotated BRIEF
PA1	Precision at 1
RAM	Random Access Memory
RBG	Red Green Blue
ROC	Receiver Operand Characteristic
ROI	Region of Interest
RTF	Ratio Test Filtering
SF	Symmetric Filtering
SIFT	Scale-Invariant Feature Transform
SPM	Selective Perspective Matching
SURF	Speeded Up Robust Features
ToF	Time of Flight
TPR	True Positive Rate
VLAD	Vector of Locally Aggregated Descriptor

Chapter 1

1. Introduction

This chapter introduces the topic of this thesis which is the extraction of distinctive features from light-fields, a rich representation format for which more reliable features can be obtained. First, the context and motivation of this work is presented, followed by the Thesis objectives. After, the structure of this thesis is described in detail.

1.1. Context and Motivation

Visual information has become incredibly relevant in modern society, and is extremely important for a growing number of applications. Initially, visual information only needed to be efficiently represented to be transmitted, stored, and later visualized, but more recently it has also become critical to efficiently represent it for other tasks, notably retrieval and recognition. In this context, visual features play a major role, as they are able to provide a concise representation of visual data that is efficient for content retrieval, and object recognition. In parallel, visual sensors have shown major developments, targeting richer acquisitions of the light in a scene. In this context, the so-called light-field cameras which have recently emerged are able to go beyond the standard acquisition models, by enriching the representation through the measurement of the light at each pixel position coming for a large set of directions. This richer light representation allows additional user functionalities, notably *a posteriori* refocusing, and perspective changing.

This work focuses on the emerging topic which combines the two technologies mentioned above: feature descriptors for light-field images. Both issues are separately very relevant in the current multimedia world, and both have been the target of extensive research in recent years, however, their combination is rather new and there is not yet much literature about it.

A feature is a piece of information derived from an image which is distinctive enough to discriminate a visual scene (or object) from others. The ability to extract meaningful and compact features potentiates a plethora of functionalities and applications such as image retrieval, object recognition, among others. In the past 10 to 15 years, there has been extensive development in this field, with the goal of creating more efficient and less computationally complex feature extraction methods. Among the most relevant local feature descriptors which came out of this research are the SIFT [1], SURF [2], and BRISK [3] descriptors. Many companies have exploited feature extraction in their applications and products, notably some of the Internet titans such as Google or Facebook. The latter has developed the DeepFace facial recognition system, while Google has developed an image

retrieval software called Google Images. However, a multitude of other image retrieval systems exist (see Figure 1).

Figure 1: Application examples. Left: The DeepFace system. Middle: Google Images. Right: Pixolution image retrieval software [4] [5] [6].

While feature description technology has been extensively developed in recent years, it has mostly targeted the popular image and video acquisition formats, notably 2D or frame based imaging and video. While 2D images and video are still the dominant visual acquisition and display format, other formats have recently gained relevance as they aim to make visual representation data closer to real light scenes. These formats may be seen as parametrizations of the plenoptic function [7], a multi-dimensional function which is able to fully represent the light in the real world. The emerging formats and associated sensors have been developed to create a more accurate visual representation of the real world, by considering not only the total light intensity at a certain position but rather its distribution according to the angle of incidence of light rays. This is done by incorporating an array of micro-lenses in the camera, thus obtaining information on millions of light-rays in a single shot. In this context, the so-called light-field cameras appeared in the market, notably developed by companies such as Lytro [8] and Raytrix [9], see Figure 2.

Figure 2: Light-Field cameras. Left: Raytrix R42 model. Middle: Lytro Illum. Right: Lytro Immerge [9] [8].

While these light-field images are an important leap forward in the world of photography, their processing is critical, as they cannot be directly presented in a conventional 2D display. Since these images contain a tremendous amount of information, the first relevant research topic is light-field image coding. The analysis of these new images in terms of feature description targeting retrieval and recognition is another major research challenge. Regarding this topic, although some research has already been made [10] [11], there is still vast room for further research.

1.2. Objectives and Structure

Following the context above, the challenge is to develop feature detection and extraction methods that exploit the light-field representation format to have better retrieval performance than other available 2D visual descriptors. In this case, the work focus on the development of local descriptors which are employed as the first step of many visual information processing systems, such as image retrieval. This Thesis has the following objectives:

- Review current feature extraction solutions, present a well-structured classification for currently existing 2D feature extraction solutions, and review in detail the most relevant solutions. Review currently existing feature extraction solutions for complex visual formats such as light-fields.
- Development of a light-field database and assessment of standard 2D descriptors when they are applied to the light-field visual representation format.
- Development of a novel 2D feature detection method that is able to outperform standard 2D feature detectors.
- Development of a novel 2D feature extraction method with two layers, one that is compliant with 2D visual descriptors and other enhancement layer that exploits the light directional information available in the light-field format.

This thesis is structured in the following way: Chapter 1 (this one) introduces the context and motivation of the thesis, and also presents its structure, Chapter 2 is dedicated to the state-of-the-art review on feature detection and extraction methods for 2D images, this chapter starts by introducing essential basic concepts such as features, descriptors, and image matching. Also, a classification framework for feature detectors and descriptors is proposed. Based on this classification, a review of the most popular solutions in this field is performed, with more emphasis on local feature descriptors, which are an essential part of this work. Chapter 3 also performs a state-of-the-art review on feature detection and extraction, but this time for plenoptic based formats, introducing first, some basic concepts on the subject such as the plenoptic function, followed by a review of the currently available solutions.

Chapter 4 performs a brief review of currently existing image retrieval datasets and presents a novel, multi layered, light-field landmark dataset captured in the city of Lisbon, containing a multitude of distinct landmarks and different image transformations; this dataset is ideal for feature detection and extraction tasks. In addition, chapter 4 presents some novel methods of light-field matching, which take into account the extra information provided by this visual format and are rather effective in terms of retrieval performance. Chapter 5 introduces a key-point detector suitable for light-field images. This feature detector exploits an alternative visual parametrization of the light-field, called the Epipolar Plane Image or EPI. This detector is heavily based on line detection and therefore makes use of the famous Hough Line Transform. This light-field detector is then assessed with a solid evaluation framework, as well as robust repeatability evaluation metrics. Chapter 6 introduces a novel feature descriptor, adapted to light-fields, which is based on the well-known SIFT [1] descriptor, and also exploits the EPI to extract more distinctive features. The proposed descriptor is evaluated according to the methods defined in chapter 4 and is compared to currently available feature descriptors, using the landmark dataset captured and described in chapter 4. This descriptor has two layers, the first layer optimizes the descriptor position by exploiting information from EPI cubes, while the second layer further exploits the structure of the light-field by computing intensity gradients along a section of an EPI cube. Finally, Chapter 7 presents some conclusions and proposes some future work to further improve the feature detection and extraction techniques for the light-field visual representation format.

Chapter 2

2. State-of-the-Art on 2D Feature Description: Detection and Extraction

In this chapter the state-of-the-art on 2D feature descriptors is presented. First, some basic concepts are introduced, next, a classification for 2D image descriptors into meaningful classes is proposed. In this classification the descriptors are divided into clusters according to their characteristics. For each cluster of descriptors, some feature detection and extraction techniques are described and compared according to its objectives, architecture, and performance.

2.1. Basic Concepts

In order to fully understand the subject of 2D feature description, some basic concepts must be briefly introduced. Since the remaining text of this chapter will eventually refer to one or more of these concepts, it is imperative to precisely define them. The first key concepts that should be defined are the following:

- **Feature:** A feature is a highly distinctive piece of information obtained from an image. A feature can be a part of an image or the entire image and allows an image to be discriminated from others.
- **Descriptor:** A descriptor is a vector that compactly describes a feature, and is the result of a feature extraction algorithm. The main focus of this chapter will be on descriptors.

Next it is important to understand the difference between feature detection, feature extraction, and image matching. The first two processing steps are used sequentially in order to obtain a descriptor, while the third uses these descriptors to find correspondences between images, Figure 3 illustrates these three steps sequentially, each step will now be briefly explained.

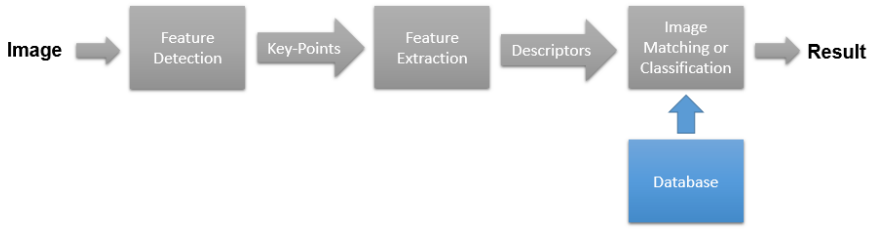

Figure 3: General architecture of an image matching or classification framework.

- **Feature detection:** Algorithm that analyses an image to obtain a subset of points, or regions, which are considered to be the most "relevant" in that image, these points are commonly referred to as key-points. The characteristics of each detected feature vary according to the feature detection algorithm being used. The scale at which each key-point is detected is usually also stored (scale spaces will be explained next). Figure 4 illustrates the results of a feature detector. Note that global descriptors (which will be described in following sections) do not apply feature detection, but rather skip directly to extraction phase.

- **Feature extraction:** This step takes as input key-points, and their respective scales, (with the exception of global descriptors which apply extraction directly), obtained during the feature detection step, and assigns to it key-point a set of characteristics that are unique, and which will be used to discriminate that same key-point from other key-points. These characteristics are obtained by processing the surrounding region of each key-point and constitute a vector which describes this region, usually with a lower dimensionality. Thus a descriptor is a vector which can have different lengths and characteristics, depending on the feature extraction method that is used.

- **Image matching or classification:** After a set of descriptors is obtained, a given key-point is considered a match to another if they present identical, or very similar, descriptors. The similarity of these descriptors is measured in different ways, depending on the method of feature description being used. A common way of measuring is the computation of the Euclidian distance between two descriptors. The process of image matching consists in running a query image through a database; note that an image has many descriptors, so if the descriptors from that query image mostly match the descriptors from another image in the database, the two images are considered a match, meaning that they represent the same scene. Figure 5 represents this matching step; notice how the images are captured from different perspectives, and yet are correctly matched.

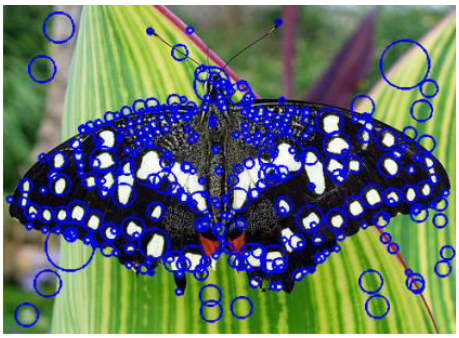

Figure 4: Example of the output of a feature detection algorithm, the radius of each circle corresponds to the scale at which each key-point was detected [12].

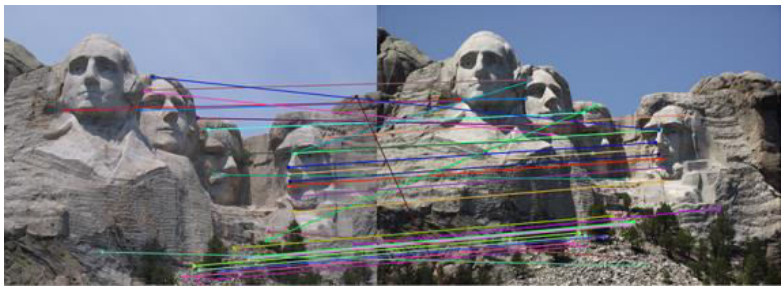

Figure 5: Example of image matching [13].

Now that feature detection and feature extraction are defined, one must consider what the ideal and desired properties of a descriptor are. In [14] a set of properties of a reliable descriptor is presented, some more intuitive than others, which are presented next:

- **Repeatability:** Repeatability refers to whether or not the same features are correctly detected, and matched, in different representations of the same scene. If a descriptor offered no repeatability, the process of image matching would almost never work, which will make a descriptor not useful.

- **Robustness to image variations:** A descriptor must be invariant by some extent to variations of the visual scene it represents. These changes can be caused by capturing an image from different perspectives, at a different time of the day, from different cameras, etc. The most typical variations are the following:

    - **Rotation or Translation:** Rotation or translation in images typically come from the acquisition of an image from different camera angles or different camera positions. It can also come from changes in the visual content of the image, for example, a moveable object may be purposefully photographed in several different positions, also causing rotation or translation in the image. Finally, image rotation can simply come from manually rotating a photograph, this is referred as in plane rotation.

    - **Lighting Variations:** Lighting variations usually come from two types of situations. Either an image was captured at different times of day or it was captured under different artificial lighting conditions which can also change.

- **Scale Changes:** To understand scale changes, it is necessary to understand the concept of scale space. A scale space is a set of images which represent the same scene, but with different levels of detail. Images can be captured at different levels of detail, so it is important that the same descriptor is obtained for images at different scales. Thus, a descriptor is able to detect an object that is close by, or further away from the camera. The scale-space is obtained by taking an image and progressively *blurring* it. Formally, this is performed by convoluting the image with the Gaussian Operator, equation (2.1) [15] mathematically represents this *blurring* process. L represents the scale space, G represents the Gaussian operator, I represents the original image, coordinates (x, y) represent the location in the image and σ represents the variance of the Gaussian operator; in practical terms, σ is the scale parameter which defines the level of *blur*. Figure 6 represents a scale-space of an image; notice how the images progressively lose detail.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \qquad (2.1)$$

  For a descriptor to be robust to scale changes, feature detection and extraction must be designed to deal with different levels of scale, so detection must identify the scales of the key-points it finds. This need for scale change robustness often implies an increase in complexity, since the same operations must be repeated for different scale levels.

- **Representativeness:** A set of descriptors for a given image must be in sufficient number, and efficiently distributed across an image to correctly describe it. If this characteristic is not guaranteed, the most important (salient) parts of the image will not be described. This means that the image retrieval or image classification tasks will not be efficiently performed (i.e. wrong matches or wrong classes are obtained).

- **Distinctiveness:** This characteristic represents the *uniqueness* of the descriptor, i.e. different features are represented as distinctively as possible. If descriptors are not distinctive, the effectiveness of the image retrieval or classification phase would be severely affected since descriptors representing different features would be similar to each other.

- **Task Efficiency:** Independently of the other properties, a descriptor must be efficient at the task at hand, meaning that it must be able to accomplish the task it was designed for, under the circumstances it was designed for. For example, a descriptor designed to be rotation invariant must have a reliable performance when dealing with rotated images.

- **Compactness:** A descriptor must occupy the fewest amount of space, i.e. to lower the high dimensionality space of an image. This must be performed without affecting its other characteristics such as repeatability, robustness, distinctiveness, etc. When a descriptor is compact, low bandwidth is necessary to transmit or store it. Besides, the feature matching step can be performed faster which is beneficial in applications with large databases.

- **Low complexity in the detection, extraction, and matching phase:** Low complexity is a target requirement for some applications such as mobile visual search or virtual reality. In the case of local descriptors, low complexity should be achieved for the detection, extraction, and matching phases.

The above properties can often be used to evaluate and compare descriptors, as such, following sections will refer to these characteristics when contrasting different descriptors.

Figure 6: Scale space example [15].

2.2. Proposing a Classification for 2D Image Descriptors

Before describing some feature detection and extraction techniques, it is important to describe the main types of detectors and descriptors which were proposed with different aims and are complementary. To classify descriptors, a classification based on what entity the features describe (e.g. the whole image, or just a region in the image), and on the characteristics of the descriptor (i.e. what is the type of each individual component of the descriptor) was followed. In addition, a feature detector classification is presented, focusing on the type of image structures. Figure 7 displays our classification.

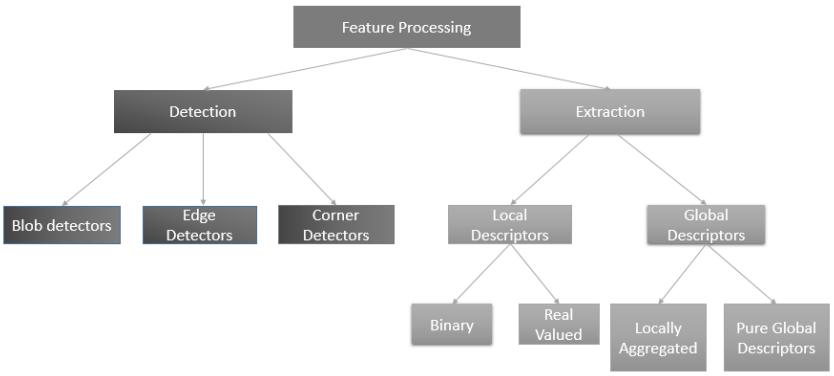

Figure 7: Classification scheme for 2D image detectors and descriptors.

As described in Section 2.1, to obtain a descriptor it is necessary to perform both detection and extraction of visual features. The feature detection and extraction techniques can be grouped into different clusters, according to their characteristics, as follows:

- **Feature Detection**: In many cases, a feature detector is the first step to describe an image. Since its main function is to search and detect salient regions of an image, a classification of detectors based on which type of regions they detect is first proposed. Thus, the following three classes are considered, each one tailored to a specific structure of the image:

  - **Blob Detectors:** Blob detectors locate regions of an image whose properties differ from its surroundings, as blob detectors do not perform detection based on geometrical properties (like other types of detectors), but rather detects its features based on properties such as colour or brightness.
  - **Edge Detectors:** As the name suggests, these detectors search for edges in images. Edges are boundaries between two different regions of an image, and therefore display strong gradient magnitude in one direction.
  - **Corner Detectors:** Corners are regions where an image seems to change in two directions, thus a corner displays strong gradient magnitudes not in one, but in two directions. The corner detectors uses this property to detect features.

- **Feature Extraction:** Feature extraction describes interest points by creating a descriptor, i.e. builds a vector of values that are informative and non-redundant. According to our classification, the descriptors can be divided into two main classes:

  - **Local Descriptors:** These descriptors use as input interest points or regions obtained by feature detection and describe each part (patch) of the image by building a vector which reflects its characteristics. Since local descriptors are applied to each interest point, a set of local descriptors for each image are usually computed to obtain a meaningful representation. Thus, multiple local descriptors are used to perform image matching and this is usually more robust as not all the descriptors need to be matched. Local descriptors can be either binary, or real valued. Meaning that they can be represented as a floating point vector, or simply as a binary string. The latter are naturally the more compact, but some (usually minor) performance losses can occur.

  - **Global Descriptors:** Global descriptors do not require an explicit feature detection step since the whole image is described. There are two types of global descriptors that differ in the way that are designed: i) aggregation of local features, i.e. multiple local descriptors are processed to obtain a global representation (where locality doesn't play any role) and ii) pure global descriptors which are usually obtained by processing the entire image (the simplest one is the colour histogram) or by segmenting the entire image with a fixed layout and jointly processing all the regions.

Table 1 and 2 show some feature detectors and descriptors correctly classified.

Detector	Blob	Corner	Edge
DoG	X		
MSER	X		
Fast Hessian	X		
CenSurE	X		
FAST		X	
AGAST		X	
BRISK		X	
ORB		X	
Canny			X
Deriche			X

Table 1: Classification of popular feature detection techniques.

Descriptor	Local	Global/Locally Aggregated	Binary	Real-Valued
SIFT	X			X
SURF	X			X
HOG		X		X
GLOH	X			X
Daisy	X			X
BRIEF	X		X	
BRISK	X		X	
ORB	X		X	
FREAK	X		X	
BoV		X		X
VLAD		X		X
GIST		X		X
Fisher Vector		X		X

Table 2: Classification of popular feature extraction techniques.

For each table, the highlighted solutions are described and compared in the following chapters. These solutions were selected due to being the most relevant in the current literature. As this report focuses more on feature extraction, descriptors will be more thoroughly explained than detectors.

2.3. Local Descriptors: Detection

Local feature detection is the first step to obtain a descriptor and is designed to locate points of an image which are considered to be relevant. Typically, the area surrounding these points (usually referred as key-points) must contain distinctive information and are stable with respect to geometric transformations and noise. Thus, a local feature detector should not only compute, relevant locations of the image, but also define a local interest area. The criteria used to locate the features depend on which type of feature detector is used. As shown in Section 2.2, feature detectors can be divided into blob, corner, or edge detectors. Blob, and corner detectors are the most relevant in current literature, so for those two classes, a few solutions are presented and briefly explained. The most relevant characteristics of feature detector is its robustness to different image transformations.

2.3.1. Blob Detectors

Blob detectors attempt to detect regions of an image that differ in properties from surrounding regions, these properties can be, for example, brightness or colour. A reliable blob detector should be able to detect these interest regions at different levels of scale, different viewpoints, and different lighting conditions. Next, the most relevant blob detectors are presented:

Difference of Gaussians (DoG)

The DoG detector [1] is used as the detector typically used by the SIFT feature extraction algorithm (see Section 2.4), and is an approximation of the Laplacian of Gaussians (LoG) filter. The LoG filter consecutively takes an image, progressively blurs it, and calculates its second order derivatives (hence the name Laplacian), and is usually applied at multiple scales to obtain a feature detector that is invariant to scale.

The LoG filter usually requires the computation of a scale-space pyramid to detect scale-space maxima/minima. This pyramid is composed by several layers, which are referred as octaves and scales, and each octave is obtained by progressively half-sampling the original image; as defined each octave has several scales (see Section 2.1 for description of a scale space). In LoG, the second order derivatives are computed for each element in the scale-space pyramid, which is computationally very demanding.

The DoG approximation reduces the complexity of LoG by exploiting the scale-space of images, and subtracting the consecutive scales (see Figure 8). The results of this subtraction are then searched for extrema using a mask which detects the most/least energetic pixel within an area of 26 neighbours, 8 in the scale corresponding to the key-point, and 9 in each of the scales adjacent to it. This approach greatly reduces the LoG complexity, since it switches from second derivative computation to simple subtraction. Additionally, SIFT also computes the exact sub-pixel position of extrema, through Taylor series expansions, and eliminates low-contrast key-points. For more detail on these last few steps, please refer to [1]. The result of this search will be the key-points which are maxima or minima of the DoG detector at each scale with an additional non-maxima suppression step. This means that each key-point is detected at a specific scale level, which is important to obtain scale invariance.

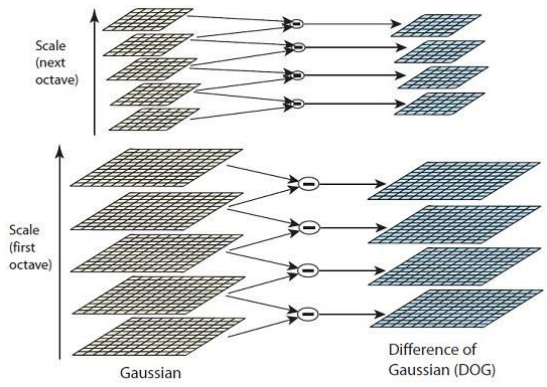

Figure 8: Difference of Gaussians approximation [15].

Fast Hessian Detector

The Fast Hessian detector [2] is also scale invariant and is typically used by the SURF descriptor (see Section 2.4) and is based on the computation of the determinant of the Hessian Matrix. The Hessian matrix describes the response of an image location to the convolution with its second order Gaussian derivatives, which logically involves considerable computational complexity. The fast hessian detector reduces this complexity by approximating Gaussian second order derivatives with box filters, which are much faster to compute. Figure 9 shows these filters.

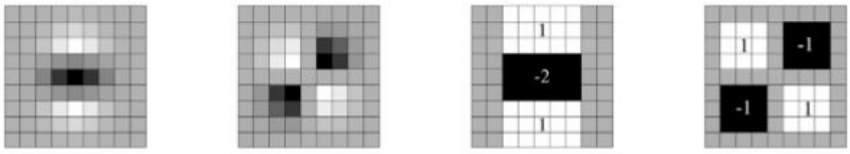

Figure 9: Left to Right: Gaussian second order derivatives in the *x*, and *xy* direction, and its corresponding box filter approximations [2].

The image is analysed throughout the scale space by up-scaling filter size, and the determinant of the Hessian matrix is computed for each pixel location as follows [2]:

$$c(x, y, \sigma) = D_{xx}(\sigma).D_{yy}(\sigma) - \left[0.9D_{xy}(\sigma)\right]^2 \qquad (2.\ 2)$$

Where $c(x, y, \sigma)$ is the Hessian matrix determinant at scale σ and pixel locations (x, y) , $D_{xx}(\sigma)$, $D_{yy}(\sigma)$, $D_{xy}(\sigma)$ are the approximated convolutions obtained through the box filters. If the determinant c is above a certain threshold it is considered a key-point, in a nutshell this computation is a way to quantify the local contrast of key-points. Furthermore, the SURF detector as in SIFT, applies a non-maxima suppression step in a $3 \times 3 \times 3$ neighborhood and eliminates low contrast key-points.

Centre Surround Extrema (CenSurE)

CenSurE [16] is a scale-invariant feature detector which also approximates the LoG filter, by using a series of centre surround bi-level filters (filters that multiply the image by 1 or -1). CenSurE is able to perform feature detection at all scale levels without half-sampling each octave, and is in that sense superior to the DoG and Fast Hessian Detector used by SIFT and SURF, since there is no need to build a scale-space pyramid for key-point detection. CenSurE also performs non-maximal and edge suppression, in a similar fashion to SIFT. The approach applied by CenSurE results in features which are more stable at higher levels of the scale-space pyramid.

2.3.2. Corner Detectors

Corner detectors aim to identify key-points by detecting regions where there are two dominant edge directions in the local neighbourhood of the point. FAST, AGAST, and BRISK corner detectors are described next.

Fast Accelerated Segmented Test (FAST)

FAST [17] is based on the Accelerated Segmented Test, which uses the Bresenham circle as a high-speed way to detect key-points. The Bresenham circle is displayed in Figure 7, this circle is composed of the pixels which lie in the circumference line of radius r, centred at a pixel p. Pixel p is considered a key-point if n continuous pixels in its Bresenham circle are all brighter than $I_p + t$, or darker than $I_p - t$, (I_p being pixel intensity value) where t is a threshold value, n is tipically 12, and r is equal to 3.

A few approaches to optimize the FAST algorithm even more are proposed in [18], one of them selects the corner points by testing only circle points 1, 9, 5, and 13 (see Figure 10), and assuming that at least three of these points must be brighter than $I_p + t$, or darker than $I_p - t$, for p to be considered a corner. Another presented suggestion, is to use a machine learning process to speed up this algorithm by creating decision trees which can classify a point as a corner or non corner with only a few pixel tests.

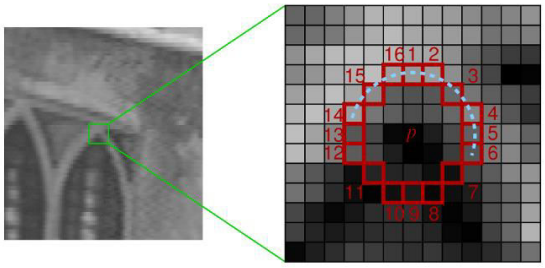

Figure 10: Bresenham circle defined using pixel c as the centre [17].

Adaptive and Generic Accelerated Segmented Test (AGAST)

AGAST [19] is based on FAST, by also using the Bresenham circle for feature detection, but enhances the FAST performance by creating a new configuration space for the decision tree (see 2. 3 [19]).

AGAST improves on its predecessor's performance (FAST) by further enhancing the decision process relative to pixel intensities. Thus, a pixel in the Bresenham circle is no longer simply labelled darker than, or lighter than, or similar to the centre pixel. The idea is as follows: choose one of the pixels in the circle to test and one question to perform. The question is then evaluated for this given pixel, and the response is used to decide the following pixel and the following question. Searching for a corner, hence, reduces to traversing a binary decision tree, instead of a ternary tree, as in FAST. The state of a pixel x regarding the centre of the Bresenham circle n , denoted as n, x is assigned as follows [19]:

$$S_{n,x} = \begin{cases} d, & I_{n,x} < I_n - t & (darker) \\ \bar{d}, & I_{n,x} \not< I_n - t \ \Lambda \ S'_{n,x} = u & (not\ darker) \\ s, & I_{n,x} \not< I_n - t \ \Lambda \ S'_{n,x} = \bar{b} & (similar) \\ s, & I_{n,x} \not> I_n + t \ \Lambda \ S'_{n,x} = \bar{d} & (similar) \\ \bar{b} & I_{n,x} \not> I_n + t \ \Lambda \ S'_{n,x} = u & (not\ brighter) \\ b & I_{n,x} > I_n + t & (brighter) \end{cases} \quad (2.\ 3)$$

Where $S'_{n,x}$ is the preceding state, I is the brightness of a pixel. Variables b, s and d refer to the state of the pixel regarding another pixel (i.e. darker, brighter, etc). And u means the state is still unknown.

To build the optimal decision tree AGAST implements an algorithm similar to the backward induction method. AGAST also performs a dynamic adaptation of the decision tree by taking into consideration that every image has homogenous and (or) cluttered areas and optimizing the decision tree according to the distribution of these two types of areas.

Binary Robust Invariant Scalable Key-points (BRISK)

Unlike FAST and AGAST, BRISK [3] performs a complete feature detection and extraction process (the feature extraction process of BRISK is described in Section 2.4). The BRISK feature detector addresses the major drawback of FAST and AGAST, by detecting key-points across a scale space, similarly to the DoG approach. BRISK applies the AGAST detector in each layer of the scale space, a point p is classified as a corner if its score s is greater than all its neighboring pixels in the same layer and in the layer above and below.

2.4. Local Descriptors: Extraction

This chapter focuses on the feature extraction of local descriptors. As stated before, a feature extraction process takes as input a key-point, its scale and (possibly) orientation, and outputs a feature vector, or descriptor, which represents the area surrounding the key-point.

2.4.1. Real Valued Descriptors

Descriptors can be divided into real valued or binary. Real valued descriptors encode image patches using real numbers with integer or floating-point precision. The correlation between real valued descriptors is measured through the Euclidian distance that defines how similar two descriptors are. The most widely used real valued local descriptors are the SIFT [1], SURF [2], and HOG [20] descriptors. The following Sections will describe and compare each one of these descriptors, as well as explain the motivation behind their design.

2.4.1.1. SIFT – Scale Invariant Feature Transform

Objectives and Technical Approach

SIFT was designed to construct a feature descriptor whose output would be invariant to scale and rotation, therefore being able to perform reliable matching between different views of an object or scene, and even to identify cluttered or partially occluded objects. Also, SIFT can also handle moderate levels of geometric distortions and changes in illumination. Despite its age (published in 2004), SIFT still remains a reference in the context of feature descriptors, displaying impressive repeatability, distinctiveness, and matching accuracy.

The SIFT descriptor is used in conjunction with a DoG blob detector (see Section 2.3), and creates real value features by dividing a patch around a key-point into 16 sub-regions. For each sub-region, an orientation is computed and placed into an 8 bin histogram. Thus, SIFT descriptors contain local visual data and are able to describe the local shape of the region using these edge orientation histograms (obtained from the gradient of the image).

Architecture and Main Techniques

Figure 11 shows the general architecture of the SIFT feature extraction algorithm, this process is repeated for every detected key-point in an image, each step will now be explained in greater detail:

Figure 11: Architecture of the SIFT feature extraction algorithm.

1. Key-point orientation assignment: To compute the SIFT local descriptor, it is necessary to receive the exact sub-pixel position of a key-point, as well as its respective scale, data usually obtained from the SIFT DoG detector. Then, for the region surrounding each key-point, a gradient magnitude and orientation is computed. The size of this region depends on the scale at which the key-point was detected. Magnitude and orientation are computed using (2. 4) and (2. 5) respectively [1]. This is done to compute the prominent orientation for the region around a key-point, and assign it to the key-point.

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2} \qquad (2.\ 4)$$

$$\Theta(x,y) = tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))) \qquad (2.\ 5)$$

$L(x,y)$ represents the luminance value of the image, at the coordinates (x,y).

2. Sub-region orientation assignment: SIFT divides the region around the key-point into 16 sub-regions, where each sub-region has a base size of 4x4 pixels, but can vary according to the scale and octave level where the key-point was detected. Gradient magnitudes and orientations are also computed for each of these sub-regions. This set of sub-regions is called a patch, and considering that SIFT is meant to be rotation invariant, two measures are taken: first, the patch is rotated according to the key-points orientation, second, the key-points orientation is subtracted from each computed orientation. Thus, each sub-regions gradient orientation is relative to the key-point for which the descriptor is being extracted.

3. Orientation Binning: Next, a histogram of orientations for each sub-region is computed. This histogram has 8 bins, each of them representing an angle range for the orientation, for example, the first bin represents 0°-44°, the second 45°-89°, and so on. This histogram accumulates, for each pixel in the sub-region, a value that depends on both the gradient magnitude and orientation.

The amount added to each histogram bin also depends on the distance from the pixel being to the key-point centre. To account for this distance, a Gaussian weighing function is used, its variance corresponds to 1.5 times the scale value for that key-point. Figure 12 displays the process of gradient magnitude and orientation assignment, and Gaussian weighing, in an intuitive way. In the left, the Gaussian Operator (the blur is the graphical representation of its 2D PDF) is shown. The grid in the middle represents the 16 descriptor sub-regions and their respective orientations, which were determined using (2. 4) and (2. 5) (the exact sub-pixel key-point position is located in the grid centre). On the right, the final result is shown, with the weighted magnitudes.

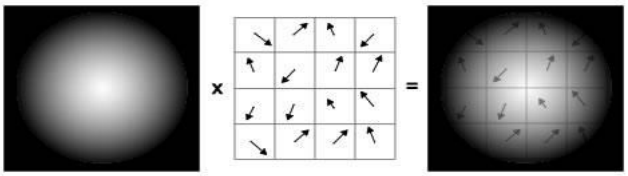

Figure 12: Magnitude and orientation assignment, followed by Gaussian weighing [15].

4. **Storing data into vector:** Finally, a vector can be created with all the data computed. For each sub-region there is an 8 bin Histogram, and there are 16 sub-regions for each key-point. Therefore, the descriptor is simply a concatenation of the 16 histograms computed for each sub-region, thus a vector of size 16×8 is obtained. Where each vector entry represents the magnitude of a given orientation bin, in a given sub-region.

5. **Vector normalization and 0.2 threshold:** To make the descriptor robust to illumination change, the vector is normalized to unit length. Therefore, the value of each vector entry is relative to the vector average value, cancelling out possible illumination changes. Non-linear illumination changes are also a factor to take into account, these can cause a large change in relative magnitude for some gradients; to counter this large change, a threshold value of 0.2 is set on the vector entries. After applying the threshold the vector is normalized once again.

2.4.1.2. SURF – Speeded Up Robust Features

Objectives and Technical Approach

SURF [2] provides a scale and rotation invariant descriptor which approximates, and at times even outperforms other descriptors in terms of repeatability, distinctiveness and robustness. Also, the descriptor computation has much lower complexity than SIFT, more precisely, SURF computation time is about one order of magnitude lower than SIFT.

The SURF descriptor is paired up with a Fast-Hessian blob detector (in Section 2.3). Similarly to SIFT, SURF also applies sub-region division, and relative orientation assignment, but instead of computing region orientation through equations like (2. 4) and (2. 5) , SURF relies on Haar wavelet transforms to compute its output. In addition, SURF also has a rotation variant version, U-SURF, which can be computed much faster, and might be ideal for applications where camera or object rotation does not occur.

Architecture and Main Techniques

Considering the excellent performance of SIFT, SURF uses some of its concepts, but obtains a technical solution with much lower complexity. Figure 13 shows the architecture of the SURF descriptor.

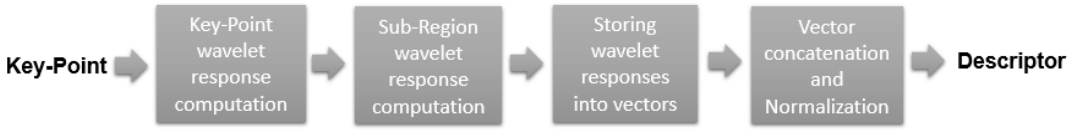

Figure 13: Architecture of the SURF feature extraction algorithm.

1. Key-point wavelet response computation: The first step is to find an orientation from a circular region around a key-point, this region has a radius equal to six times the scale value at which the key-point was detected. For this purpose, the Haar-wavelet responses in the x and y direction of the circular region are computed. Figure 9 shows the Haar-wavelet types used for SURF. Haar-wavelet types are used as a means to determine in which directions an image patch can vary, if a path is homogenous, it should be "unresponsive" to any Haar-wavelet type, if it displays changes in the x direction it should be responsive to the upper operator in figure 15. These responses are subjected to Gaussian weighing centred at the key-point (see figure 16), and the dominant orientation is found by computing the sum of all responses within an orientation window of 60° degrees, as shown in Figure 14. Note that for U-SURF, this step is skipped, since the relative orientation assignment is not performed.

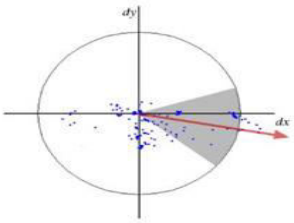

Figure 14: Computation of dominant region orientation in SURF [21] .

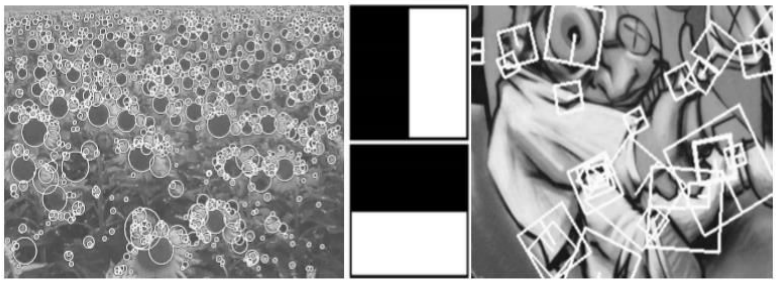

Figure 15: Left: Detected interest Points in SURF (using Hessian-based detectors). Middle: Haar Wavelet types used for SURF. Right: Identified rotated square regions [2].

2. Sub-region wavelet response computation: Now that the key-point orientation is known, the second step is to construct a square region around the key-point and compute a descriptor for that patch. This region is rotated according to the orientation previously computed. The patch size is equal to twenty times the scale value at which the key-point was detected. Figure 15 shows the oriented patches for each key-point detected. The patches are divided into smaller 4×4 square sub-regions, each sub-region is sampled 5×5 times (see Figure 16) and Haar wavelet responses are computed for each one, using the two Haar wavelet types in Figure 15.

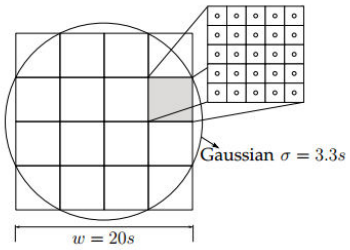

Figure 16: A 20s (twenty times the scale value) area is divided into 4x4 sub-regions, each sub-region is sampled 5x5 times to get the Haar wavelet response [21].

3. Storing wavelet responses into vectors: The following vector is used to store Haar wavelet responses in each direction, for each sub-region [2].

$$v = \{\sum dx, \sum |dx|, \sum dy, \sum |dy|\} \qquad (2.6)$$

Elements dx and dy are the Haar-Wavelet responses to the horizontal and vertical operators in figure 9. The vector entries $\sum dx$ and $\sum dy$ represent the weighted sum of all the wavelet responses over the x and y directions of each sub-region. The absolute values $\sum |dx|$ and $\sum |dy|$ are also stored. The wavelet responses over the x and y directions are always computed in relation to the key-point orientation, and are also subjected to Gaussian weighing centred at the key-point. Figure 17 shows a few examples of the construction of a v vector.

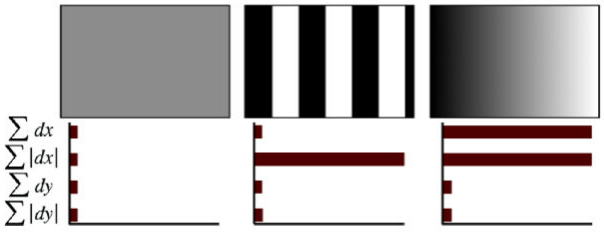

Figure 17: Descriptor entries for one sub-region. Left: For a homogeneous regions, all entries are low. Middle: In presence of frequencies in the x direction, the value of ∑|dx| is high. Right: If the intensity is gradually increasing in the x direction both ∑|dx| and ∑dx are high [2].

4. **Vector Concatenation and normalization:** Next, the SURF descriptor for a key-point is built by concatenating the 16 v vectors of each sub-region, which leads to a vector with 64 entries. Finally, the descriptor is normalized so that each vector entry is only relative to the average vector value to achieve invariance to illumination changes. Alternatively, a SURF descriptor with 128 size vector

can be created, which computes the sums $\sum dx$ and $\sum |dx|$ separately for $y < 0$, and $y > 0$, and logically does the same for $\sum dy$ and $\sum |dy|$.

2.4.1.3. HOG – Histogram of Oriented Gradients

Objectives and Technical Approach

The HOG [20] feature descriptor was designed with the purpose of object recognition in difficult scenarios such as cluttered backgrounds, or difficult illumination. HOG was originally designed for human detection in static images, but has since been extended to video, vehicle and animal detection in static images. In a similar way to the SIFT descriptor, the main idea is that object appearance and shape can be well described with the distribution of intensity gradients or edge directions. However, while SIFT does this for specific interest points, HOG uses a grid of uniformly spaced cells.

HOG models the local object appearance by dividing the image into local regions called cells, and for the pixels within each cell a histogram of gradient directions is constructed. HOG can be used for both colour and grayscale images and the descriptor is obtained by concatenating the histograms of orientations for every cell.

It is also important to note that HOG is not a usual local descriptor, since it describes not a small region, but the entire image. Therefore HOG does not require key-points as input, and thus can be used on its own, whereas SIFT and SURF both require previous key-point detection. While some might argue that HoG is a global descriptor, since it describes an entire image, HoG's method involves the extraction of several local features, and the concatenation of such features, so in that sense, HoG is not a pure global descriptor, since it does not describe the image as a whole, but instead describes small regions in the image, and concatenates them, fitting the definition of locally aggregated descriptors. However, since a considerable part of the development phase of the thesis could possibly be based on gradient computation, we wish to describe HOG in greater detail than other global descriptors presented in Section 2.5.

Architecture and Main Techniques

Figure 18 shows the general architecture of HOG, for simplicity, we will follow the specific methodology applied by Dalal and Triggs in [20] to better understand the algorithm.

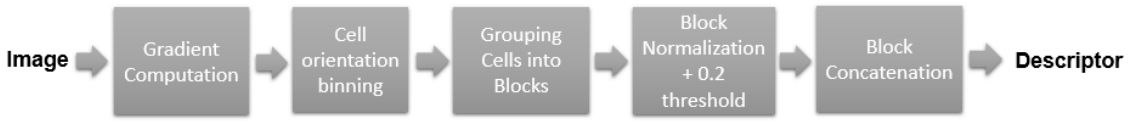

Figure 18: Architecture of the HOG feature extraction algorithm.

1. **Gradient Computation:** HOG's first step is gradient computation, this is done by applying a 1-D discrete derivative mask in both the horizontal and vertical directions of an image, $[-1,\ 0,\ 1]$ and $[-1,\ 0,\ 1]^T$ are the horizontal and vertical kernels, respectively. Figure 19 shows the graphical results of this process:

Figure 19: Left: Original grey scale image Right: Gradient Magnitudes for the same image [22].

2. Cell orientation binning: HOG then divides an image into a uniform cell grid, the cells can either be rectangular or circular, although in [20] the optimal cell dimension was found to be rectangular 8x8 pixel cells. For each cell, an orientation based histogram is created, where each pixel cast a weighted vote for this histogram, this vote depends on the gradient magnitude at the pixel coordinates. Histograms can either be unsigned, ranging from 0° to 180° degrees, or signed, ranging from 0° to 360° degrees. In [20] an unsigned, 9 bin histogram was used.

3. Grouping Cells into blocks: The cells are then grouped into larger special entities called blocks, this is done to normalize local gradient strengths, in order to compensate for changes in illumination and contrast. These blocks can vary in dimension and can either be rectangular or circular; they also typically overlap, meaning that two adjacent blocks may contain the same cell. In [20], 16×16 blocks of four 8×8 pixel cells are used. The histogram values of the cells inside each block are concatenated into a vector.

4. Block Normalization, 0.2 threshold and block concatenation: The vectors for each block must now be normalized. In [20], Dalal and Triggs explored several methods of normalization, and the selected solution is called L2-Hys, which first applies (2.8), and subsequently limits the values of v' (the normalized vector) to 0.2, and renormalizes, just as in SIFT.

$$v' = v/\sqrt{v/(\|v\|_1 + \varepsilon)} \qquad (2.7)$$

v is a unnormalized descriptor vector, and ε is a small constant. The final descriptor is simply the concatenation of all the normalized block vectors.

To apply this descriptor to object recognition, one must feed it to a recognition system based on supervised learning, the support vector machine (SVM) is one of the most popular choices regarding recognition systems. Once an SVM has been trained in images containing a particular type of object, it can decide on the presence of that same object in other images; this object can be a human being, or any other discernible shape.

2.4.2. Binary Descriptors

A more compact, and often computationally simpler alternative to real valued descriptors are binary descriptors. In these, the vector that characterizes a feature is composed of binary elements, which represents a considerable increase in compactness. Also, when descriptors are matched a reduction

in computational complexity is obtained since two descriptors can be compared by simply computing the Hamming distance between them. This Hamming distance computation is just the result of the XOR bitwise operation between two descriptors and then summing all the elements of this difference, making it much faster than Euclidian distance computation. This section describes and evaluates BRIEF [23], BRISK [3], and FREAK [24] , three of the most relevant binary descriptors.

2.4.2.1. BRIEF – Binary Robust Independent Elementary Features

Objectives and Technical Approach

The purpose of BRIEF [23] is to compute a reliable binary descriptor with comparable efficiency to real valued descriptors such as SIFT or SURF, but having much lower complexity. Now, the descriptor is a binary string, and each individual entry is obtained through pair-wise intensity comparisons on smoothed image patches; these pairs of locations are selected according to a pre-determined spatial distribution and are usually referred as sampling pattern.

Architecture and Main Techniques

The BRIEF descriptor architecture is quite simple, as displayed in Figure 20 and is repeated for every key-point detected.

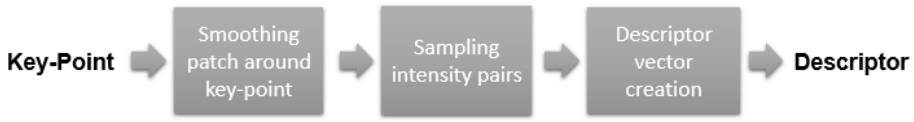

Figure 20: Architecture of the BRIEF feature extraction algorithm.

The BRIEF descriptor consists in the following steps:

1. **Smoothing patch around key-point:** The first step is to filter the image patches that BRIEF receives as input. Since all intensity comparison tests performed by BRIEF are with respect to two single pixel values, these tests are very sensitive to noise. Thus, the patch is smoothed before these tests are performed, which increases the stability of the descriptor. For the BRIEF binary feature descriptor, Gaussian smoothing kernels are applied over each image patch; each patch is a 9×9 pixel window. The standard deviation of Gaussian Kernels can assume values between 1 to 3 which leads to good performance.

2. **Sampling intensity pairs:** The next step is to sample the pairs used for the pair-wise intensity comparisons. The distribution used for this sampling is determined offline, in [23] different sampling patterns were evaluated, but a Gaussian $\left(0, \frac{1}{25}S^2\right)$ distribution achieved the best results in terms of recognition rate.

3. **Descriptor vector creation:** To create the BRIEF descriptor, a binary vector is computed, each element (bit) of this vector is obtained by performing an intensity test, for every pair of locations previously defined. In this step, the BRIEF feature vector is computed as follows [23]:

$$\tau(\boldsymbol{p}; x, y) = \begin{cases} 1 & if\ \boldsymbol{p}(x) < \boldsymbol{p}(y) \\ 0 & otherwise \end{cases} \qquad (2.\ 8)$$

Where $\boldsymbol{p}$ represents an image patch, $\boldsymbol{p}(x)$ represents pixel intensity at location x, x and y are locations in the patch represented by two dimensional coordinates (u, v). The number of location pairs x, y defines the size of the descriptor. The full binary descriptor is represented by [23]:

$$f_{nd}(\boldsymbol{p}) = \sum_{1 \le i \le nd} 2^{i-1}\, \tau(\boldsymbol{p}; x_i, y_i) \qquad (2.\ 9)$$

Where nd represents the number of intensity comparison tests performed, and thus also represents the descriptor size.

2.4.2.2. BRISK – Binary Robust Invariant Scalable Key-Points

Objectives and Technical Approach

BRISK [3] aims to provide a reliable and robust descriptor, while achieving low computational requirements. Inspired by BRIEF, BRISK also offers a low computational effort, but is more reliable, since it is invariant to some image transformations such as scale changes and rotations, for which BRIEF doesn't provide a satisfactory performance. The approach followed by BRISK aims to find a new balance between robustness and complexity.

BRISK is scale invariant since its detection phase performs scale space feature detection to obtain key-points across several scale levels. BRISK also performs orientation estimation, and uses a new, deterministic sampling pattern for image patches. By enhancing BRIEF with a new set of tools, BRISK can compute more reliable descriptors while still maintaining complexity levels low enough for real-time applications.

Architecture and Main Techniques

The key-points obtained by BRISK must be detected using a feature detector that estimates the scale of each key-point, for this purpose, BRISK uses its own detector, which performs AGAST feature detection across a scale space. For more detail on detection check Section 2.3. Figure 21 displays the BRISK architecture.

Figure 21: Architecture of the BRISK feature extraction algorithm.

The main steps of BRISK are the following:

1. Smoothing patch around key-point: Just as BRIEF, the patch around each detected key-point must be smoothed in order to reduce the intensity pairs' sensitivity to noise. But while BRIEF applies the same smoothing kernel uniformly across the entire patch, BRISK applies different smoothing kernels, according to the position of the sampling point with respect to the key-point. Figure 22 represents the sampling locations and smoothed patches, the dashed red circles represent the standard deviation **σ** of the Gaussian Kernel used to smooth the intensity values around the sampling locations [3]. This smoothing process is applied to reduce sensitivity to noise when sampling image intensities.

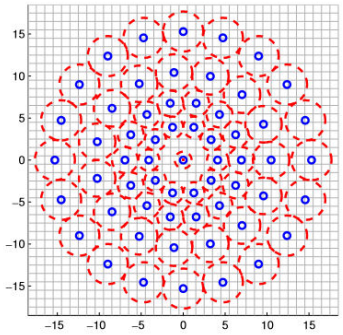

Figure 22: BRISK sampling pattern [25].

2. Sampling intensity points: After smoothing the intensity values according to the pattern in Figure 22, the new, smoothed intensity pairs are sampled. The small blue circles in Figure 22 are the exact sampling locations.

3. Rotating sampling pattern: Each patch must be rotated according to the orientation of the key-point it represents. To compute the orientation around a key-point the new smoothed intensity values are used to estimate local gradients. Consider $\boldsymbol{p_i}, \boldsymbol{p_j}$ as sampling point pairs, the local gradient $g(\boldsymbol{p_i}, \boldsymbol{p_j})$ is estimated as follows [3]:

$$g(\boldsymbol{p_i}, \boldsymbol{p_j}) = (\boldsymbol{p_i} - \boldsymbol{p_j}).\frac{I(\boldsymbol{p_j}, \sigma_j) - I(\boldsymbol{p_i}, \sigma_i)}{\|\boldsymbol{p_j} - \boldsymbol{p_i}\|^2} \qquad (2.\ 10)$$

$I(\boldsymbol{p_j}, \sigma_j)$, is the image intensity at location $\boldsymbol{p_j}$, and scale σ_j [9]. After this, short-distance and long distance pairs are determined by applying distance thresholds $\delta max = 9.75t$ and $\delta min = 13.67t$, where t is a scale value of the key-point, between every possible pairwise distance. If a distance between two points is below δmax, they are considered a short distance pair, if it is above δmin, they are considered a long distance pair. Only the long distance pairs are used to estimate key-point orientation, which are more reliable. The overall orientation estimation for each key-point is the following [3]:

$$\boldsymbol{g} = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{L}.\sum_{(\boldsymbol{p_i}, \boldsymbol{p_j}) \in \boldsymbol{\alpha}} g(\boldsymbol{p_i}, \boldsymbol{p_j}) \qquad (2.\ 11)$$

where $\boldsymbol{L}$ is the number of long distance pairs used while $\boldsymbol{\alpha}$ is the subset of the long distance pairs. The sampling pattern is then rotated according to $\alpha = \arctan(g_x, g_y)$ around the key-point. The

sampling pattern for BRISK is applied only to short-distance pairs; by doing this, both complexity and descriptor size are reduced.

4. Creating descriptor vector: To create the descriptor vector, binary intensity tests are performed over all the selected short distance pairs. The intensity test comparison used to build a BRISK descriptor is as follows:

$$\boldsymbol{b} = \begin{cases} 1 & if\ I(\boldsymbol{p_j}^{\alpha}, \sigma_i) > I(\boldsymbol{p_i}^{\alpha}, \sigma_i) \\ 0 & otherwise \end{cases} \qquad (2.\ 12)$$

The BRISK descriptor is therefore a concatenation of every intensity test result, and a final 512 bit, or 64 byte descriptor is obtained.

2.4.2.3. FREAK – Fast Retina Key-Point

Objectives and Technical Approach

FREAK [24] is one of the most recent feature descriptors, and was designed to provide a descriptor which would be able to be computed for devices with low memory and with low computational complexity resources. Despite its low target complexity, FREAK was also designed to provide robustness to scale, rotation and noise. FREAK follows the general methodology of BRIEF and BRISK by computing binary descriptors through a sequence of pair-wise intensity tests, but innovates in the way that it defines the sampling pattern (i.e. the distribution of its binary test locations); the sampling pattern used by FREAK is inspired by the retinal receptive fields, which are distinctive areas of the human retina in which external stimulations lead to different neural responses.

Architecture and Main Techniques

Figure 23 shows the architecture of FREAK, it follows the same generic procedure as its preceding binary descriptors.

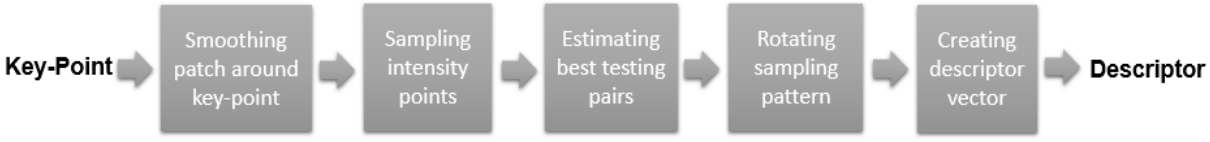

Figure 23: Architecture of the FREAK feature extraction algorithm.

The main steps of FREAK are the following:

1. Smoothing patch around key-point: Like all other binary descriptors, FREAK first applies smoothing kernels to the area around each key-point, and like BRIEF, the distance from the sampling point to the key-point determines the standard deviation of the Gaussian kernel used for smoothing. In the case of FREAK, the standard deviation increases exponentially as the distance to the key-point increases.

2. Sampling intensity points: FREAK takes the area around each key-point, and applies a sampling pattern similar to the retinal ganglion cells distribution in the human eye; this results in a non-uniform sampling grid. Figure 24 shows this sampling pattern.

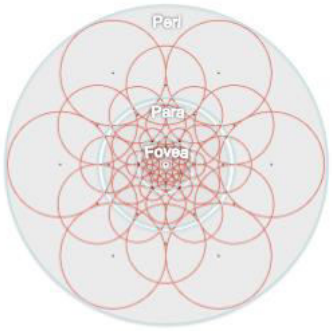

Figure 24: FREAK sampling pattern [25].

Each point represents a sampling location, while the radius of the circle around each point is the standard deviation of the Gaussian Kernel used to smooth the intensity values around each sampling point. Peri, Para, and Fovea represent three areas of the human retina.

3. Estimating the best intensity pairs: Since the sampling pattern defines a significant number points, a very large number of possible combinations for the intensity tests (that require a pair of points) are possible. To address this issue, FREAK searches for the sets of pairs which have the lowest correlation between them, and sorts these pairs into 4 different clusters of 128, each cluster represents a different level of correlation, obtaining a total set of 512 pairs. It was found that clusters after the first 512 pairs were not useful [24].

The idea behind these different clusters of pairs is to apply a coarse-to-fine analysis in the matching step, where the first 16 bytes (or 128 bits) of two descriptors are compared to search for a match: if their distance is larger than a certain threshold, they are automatically labelled as mismatches, and the rest of the elements of the descriptors are not compared. Otherwise, the next cluster of pairs is compared, and the process continues iteratively. This greatly reduces the matching time, since 90% of matches can be discarded just with the first 16 bytes. Figure 25 illustrates these 4 clusters. FREAK is therefore a 512 bit or 64 byte descriptor.

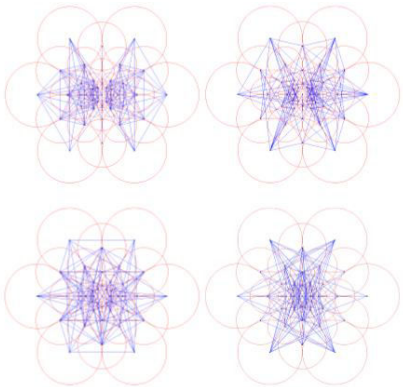

Figure 25: Different clusters of testing pairs for FREAK [25].

4. Estimating and applying key-point orientation: To estimate key-point orientation, FREAK computes a sum of local gradients over a few selected pairs, and rotates the sampling pattern according to it. While the BRISK descriptor uses long pairs to compute orientation (see Section 2.4.2.2), FREAK chooses intensity tests (pairs of points) with symmetric Gaussian kernel, as in Figure 26.

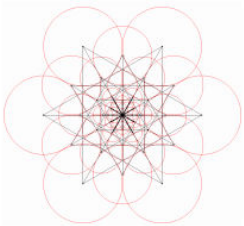

Figure 26: Pairs selected to estimate key-point orientation [25].

To compute the orientation FREAK uses only 45 intensity tests, which is lower than the hundreds of pairs used by BRISK. The orientation is computed as [24]:

$$O = \frac{1}{M}\sum_{P_0\,\varepsilon\,G}(I(P_0^{r1}) - I(P_0^{r2}))\,\frac{P_0^{r1}-P_0^{r2}}{\|P_0^{r1}-P_0^{r2}\|} \qquad (2.\ 13)$$

Where M is the number of selected pairs, and P_0^{r1} is a 2D vector containing the spatial coordinates of the center of the Gaussian Kernel. G is the set of all pairs used to compute orientation. $I(P_0^{ri})$ is the intensity value at a given location.

5. Computing and storing binary tests: FREAK computes the intensity tests in the same way as BRIEF or BRISK, meaning that if the intensity of a sampling location is higher than the intensity of its corresponding pair, the intensity test result is 1, otherwise, the result is 0. Thus, the FREAK descriptor is represented as binary vector.

2.4.3. Local Descriptor Performance Assessment

This section will analyse the performance of the different local descriptors presented in this chapter, descriptors will be evaluated using image datasets that are designed to include several transformations such as viewpoint changes, brightness changes, rotation, noise, blur, and compression.

Descriptors can be evaluated in terms of computational complexity, as well as robustness to image transformations. For the latter, different methods exist to evaluate the descriptors performance, but one of the most used is the Receiver Operating Characteristic, or ROC curve. The ROC is created by plotting the true positive rate (TPR) against the false positive rate (FPR), the true positive rate measures the percentage of correct matches (a correct match occurs when the same feature is correctly detected in different images) over the total number of matches, while the false positive rate is the percentage of incorrect matches. The better the performance of the descriptor, the larger the area under the curve. Figure 27 shows a side by side comparison of different feature descriptors using two different datasets, and two different feature detectors (DoG and Harris detectors) ROC curves were

traced for each detector-dataset combination. By analysing each combination, under these datasets, SIFT is the most reliable feature detector. Regarding the binary descriptors, BRISK and Oriented BRIEF (ORB) seem to perform best, for detail on ORB, please refer to [26].

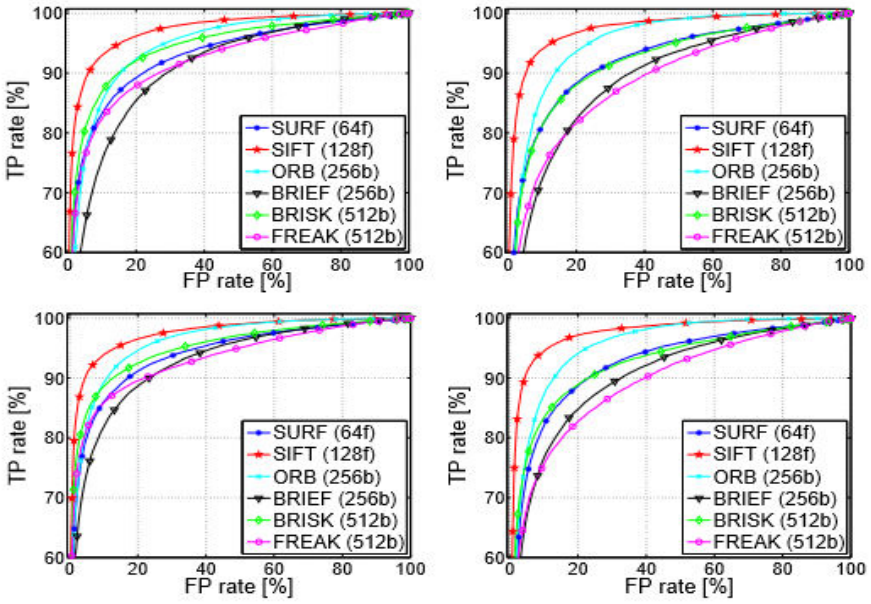

Figure 27: ROC curves for, from left to right, top to bottom: Liberty dataset, using DoG detector, Liberty dataset using Harris detector, Notredame dataset using DoG detector, and Notredame using Harris detector [27].

Table 1 displays a comparison of different feature descriptors with respect to the average processing time of 500 descriptors (i.e. the average time each method takes to create 500 descriptors). The experiments were carried on a computer powered by an Intel i7-3770 CPU at 3.40GHz, with 8GB of RAM and the processing tests only use a single core. Note that despite SIFT being the best performing descriptor in terms of matching rate (as shown by Figure 27), it is by far the most complex. Also worth mentioning is the fact that any binary descriptor is at least one order of magnitude faster than the real valued descriptors.

Descriptor	Proc. time
SIFT	43.45 ms
SURF	13.43 ms
BRIEF	1.43 ms
ORB	1.36 ms
BRISK	2.11 ms
FREAK	1.09 ms

Table 3: Average processing time for different feature descriptors [27] .

2.5. Global Descriptors

While local descriptors describe an image patch, i.e. a region of the image, pure global descriptors process the whole image to obtain a meaningful representation without any localization associated to each element. However, global descriptors may use local descriptors as input, and then further

process all local descriptors to obtain one global descriptor. A few relevant state-of-the-art global descriptors will now be briefly presented.

Bag of Visual Words (BoV)

The bag of visual words feature descriptor [28] [29] is based on the Bag of Words concept [30] often used in text processing. In that model, a text, such as a sentence or document is represented as a set of words (bag), regardless of word order or grammar, but taking into account word frequency (i.e. how many times each word appears in the text).

In BoV each image is treated as a set of local patches (after the feature detection), where these patches can be described by local descriptors such as SIFT, which converts a patch into a 128 real valued descriptor (vector). BoV converts this set of local descriptors into a set of codewords, this process is done offline. A codeword is a representation of several similar descriptors (each one associated to an image patch). BoV groups all descriptors of an image into clusters of codewords by using the method of k-means clustering. K-means clustering [31] is a popular method used for cluster analysis, which aims to partition n observations into k clusters, in which each observation belongs to the cluster with the nearest mean. The codeword is defined as the centre of a learned cluster. A codebook is defined as the set of codewords.

Each descriptor in an image is mapped to a certain codeword through the clustering process, and the entire image is represented through a histogram of codewords, (i.e. how many times each codeword appears in the image), thus forming the BoV descriptor. In order to search for the closest match to a query image, BoV searches for the image in the dataset with the most similar (according to some metric) histogram, as in Figure 28.

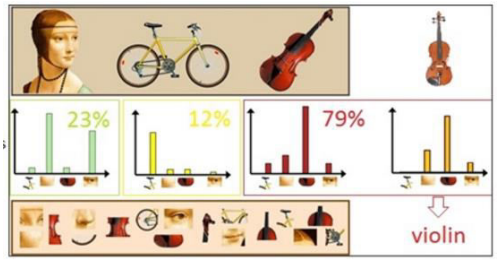

Figure 28: Bag of visual words processing, dataset is searched for a match to a query image containing a violin [32].

Vector of Locally Aggregated Descriptors (VLAD)

Another alternative to global feature description is the VLAD [33] descriptor, which displays many similarities to the BoV global descriptor, like the use of clustering, and the use of local descriptors.

First, a codebook of visual words (typically with vocabulary of size 64 or 256) is precomputed offline, through k-means clustering. Each input image is then processed by using a feature detector, and described using a local feature descriptor, such as SIFT. Each descriptor is assigned to its nearest visual word in the codebook. For each of the precomputed visual words, the vector distances between them and the descriptors assigned to them are computed, and stored into a vector. The VLAD descriptor is the concatenation of all vector distances to the k words, thus obtaining a $D = d \times k$

sized global descriptor, d being the local descriptor size, and k the size of the codebook, an element $v_{d,k}$ of the VLAD descriptor is as follows:

$$v_{d,k} = \sum x_d - c_{k,d} \qquad (2.\ 14)$$

Where x_d is the dth element of the local descriptor, and $c_{k,d}$ is the dth of its corresponding visual word. Thus, while BoV compares different images by analysing their histograms of codewords, VLAD searches for the nearest match through the numerical difference between descriptors and cluster centre.

Chapter 3

3. State-of-the-Art on Plenoptic Based Feature Description: Detection and Extraction

This chapter focuses on feature description methods for plenoptic based visual representation, namely feature detection and extraction for plenoptic representations such as the ones obtained from light-field cameras (e.g. Lytro). This a very interesting topic since plenoptic representations provide much more information than standard visual formats. The extra information that a plenoptic representation format provides can be exploited to create more efficient feature detectors and extractors. The structure of chapter 3 is similar to chapter 2, with the exception of the classification section since there is not much work available in the literature and the grouping of solutions into meaningful clusters is a very difficult task. Basic concepts about plenoptic visual representations are introduced first. After, a brief review of the few plenoptic based feature detection and extraction solutions is performed.

3.1. Basic Concepts

To understand any topic related to plenoptic data it is necessary to understand first the plenoptic function, which is the starting point for any computer vision related topic.

The Plenoptic function

The plenoptic visual representation is a way to represent visual reality that surrounds us. This representation is based on the plenoptic function, which describes all visual information in the world. This function is arguably one of the most important concepts to grasp in this Chapter.

All the visual information in the world is ultimately represented by a description of the electromagnetic field in the visible spectrum for every point in space [34]. To provide a function which quantifies this, a few assumptions must be made. The first being assuming only non-coherent light. Non-coherent light sources emit light with frequent and random phase changes between photons, while in coherent light sources the photons are all in phase.

By assuming this, a propagating wave can be described as an infinite sum of random-phase sinusoids, each with different energy, which is possible to represent mathematically as a power density for each wavelength λ. Following this reasoning, an electromagnetic wave at any point (x, y, z) in space can be represented as a sum of wave fronts coming from all directions, each direction described by an azimuth, orientation pair (θ, Φ). If a wave's energy varies in time, this can be

quantified by adding in a temporal dimension. Putting all this information together it is possible to obtain the plenoptic function as $(x, y, z, \Theta, \Phi, t, \lambda)$. In Figure 29 this function is conceptually visualized.

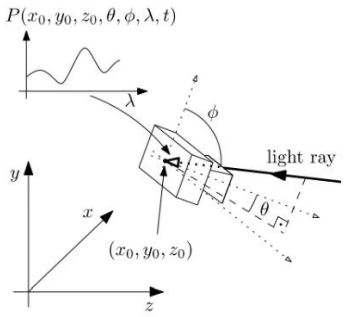

Figure 29: Visual representation of the Plenoptic Function [34].

Plenoptic function reductions

Naturally, being a 7 dimensional function, the plenoptic function is computationally very heavy to compute and to visually represent and thus it is necessary to reduce its dimensionality. Ever since the 19th century that technology found ways to sample with a lower dimensionality the plenoptic representation, from black and white photography, to 3D cinema, and everything in between. In this sense, every visual representation of the real world that has ever been conceived is a reduction of the plenoptic function. While some are simpler and easier to understand, such as photography; where the plenoptic data is just a 2D (x, y) plane, or video; where the temporal dimension is introduced, creating a (x, y, t) representation, others are more complex and richer in information. These will be listed and presented in this section, as they are essential to understanding the rest of the report.

- **Light-Fields:** Light-fields are a 4D parametrization of the plenoptic function, in current literature light-fields are divided into several categories, the most relevant of which are Multiview light-fields, and lenslet light-fields, the main difference between these two formats is the way they are captured. The two will now be explained:

  - **Multiview Light-Fields, and Multiview + Depth:** Multiview are captured using an array of cameras. These cameras can be sorted in a multitude of arrangements, the simpler being 1D linear or circular arrays of equally, or irregularly, spaced cameras. A more complete alternative in terms of data acquisition is to use 2D arrays of cameras, which can also be equally, or irregularly spaced. In this case, the Plenoptic function is represented in four dimensions $P(k, l, u, v)$, variables (k, l) are the position of the camera in the 2D array, while (u, v) is the position where a light ray strikes the sensor. In the case of Multiview video, time is included as a 5th dimension, originating a 5D $P(k, l, u, v, t)$ plenoptic reduction, so for video using RGB, we need three 5D functions $P_R(k, l, u, v, t)$, $P_G(k, l, u, v, t)$, and $P_B(k, l, u, v, t)$. Figure 30 shows an example Multiview capture.

Figure 30: Multiview example [34].

Another way to include extra information is to acquire depth information, that is, the distance from the camera plane to the surface point that reflects the light ray. This way an additional $d(k,l,u,v,t)$ depth function is added to three RGB plenoptic functions. This is referred to as Multiview + depth.

- **Lenslet Light-Fields:** Lenslet light-fields follow a very similar parametrization process to Multiview light-fields. However, unlike Multiview, a lenslet light-field is captured using a single, wide aperture camera. In this case the plenoptic function is reduced to a two plane parametrization, plane uv is the plane of the camera lens, while plane st is the plane of the sensor itself, this way any light ray that the camera captures is simply the two sets of coordinates, one pointing to where the ray intersects the lens, and the other to where it hits the sensor. Figure 31 shows this two plane parameterization.

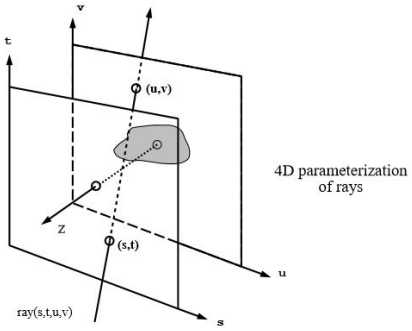

Figure 31: Two plane Parametrization of light rays [35].

- **Point-Clouds and Mesh Models:** Unlike the other plenoptic representations that are here presented, point-clouds are meant not to represent light-rays, but actual three dimensional objects. Point-clouds represent objects by associating a colour value to a given position (x,y,z), originating three 3D functions $P_R(x,y,z)$, $P_G(x,y,z)$, and $P_B(x,y,z)$, for an RGB representation. To include information on the direction of light-rays in this model it is possible to take into consideration the direction of light-rays hitting each point, obtaining a 5D $P(x,y,z,\theta,\Phi)$ function. An alternative to point-cloud representation that is also object oriented are the mesh models. In a mesh model an object is represented not by points, but by a set of connected polygons, bringing extra information about the shape of an object, the colour of each polygon is obtained by interpolating from the colour of the vertices. Figure 32 shows a point-cloud, and a mesh model example.

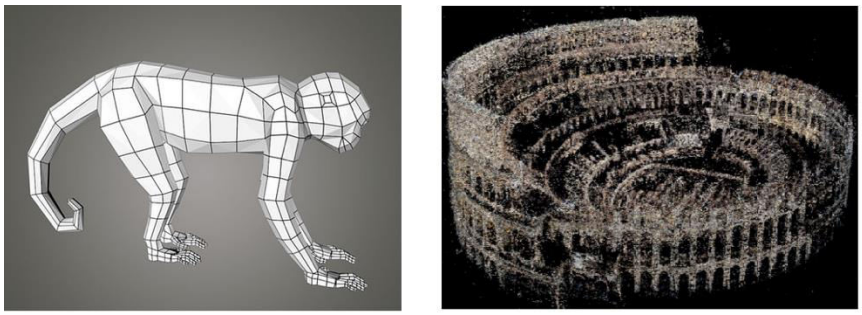

Figure 32: Left: Mesh Model. Right: Point Cloud Model [36] [37].

Acquiring Plenoptic Data

As stated before, several technologies allow sampling plenoptic data with lower dimensionality. The most traditional ways to sample plenoptic data are already well known and don't require much explanation in the context of this report, such is the case of traditional photographic and video cameras. However, since the dimensionality reductions of the plenoptic function that were above presented are quite recent and conceptually complex, some alternatives for capturing this type of data are enumerated next:

- **Arrays of Traditional cameras:** These arrays can be used to capture multiview content; arrays can be setup in several ways, including 1D linear or circular arrays, or 2D arrays of cameras. These can either be equal, or irregularly spaced, depending on the application.

- **Arrays of Micro lenses:** These arrays are in fact a single photographic camera, popularly known as light-field cameras. The Lytro company has developed several models of micro lens cameras in recent years. The cameras capture information on the direction, colour, and intensity of millions of individual rays of light, which allows for *a posteriori* photograph refocusing, and also slight perspective shifts. See Figure 33.

Figure 33: Left: Lytro Cameras. Right: Example of refocusing for "light-field" images [38] [39].

- **360° static or video cameras:** As the name suggests, 360° cameras capture an all-around view of the scene. Usually being spherically shaped, these cameras have sensors spread all over their structure which take enough data to cover the entire "visual sphere" that surrounds them. Figure 34 shows two models of 360° cameras.

- **Depth capturing cameras:** There are multiple alternatives when it comes to capturing the depth of a scene, LIDAR (a combination of the terms "light" and "radar") cameras are one of these

alternatives. A LIDAR camera records depth by illuminating objects with laser pulses and analysing the reflected light. Other depth analysing devices include Microsoft's Kinect, and ToF (time of flight) cameras. See Figure 34.

Figure 34: Left: Bubl and Panono camera models. Right: ToF Camera [40] [41] [42].

3.2. Solutions for Plenoptic Based Feature Detection and Extraction

In chapter 2 various solutions for feature detection and extraction in 2D images were presented. In this chapter the same approach is followed. But while there are numerous solutions for 2D feature detection and extraction, there is a very limited number of solutions for plenoptic based detection and extraction. As such, the solutions presented next are fewer, and less diverse in terms of technical approach.

3.2.1. 3D Key-Point Detection by Light-field Scale-Depth Space Analysis

Objectives and Technical Approach

In [11] a method for key-point extraction in light-field images is presented. While there are many ways to extract key-points for 2D images, for light-fields, the work described here is the only method of key-point extraction found. This method has received the best paper award at the International Conference on Image Processing in 2014 and detects key-points by computing a scale-space for light-field images. After, an edge detector is applied to each layer of the scale space. Scale-space construction is based on the usage of a modified version of the Gaussian kernel (which is widely used for 2D scale-space construction), called by the authors the Ray-Gaussian kernel.

Architecture and Main Techniques

Figure 35 shows the architecture of this feature detector:

Figure 35: Architecture of the Light-Field feature detector.

The main steps of this detector are the following:

1. **Creating Stack of Perspective images:** The first step is to take a raw light-field image and de-multiplex it into a set of perspective images. Each of these images captures a different perspective,

i.e. a certain direction of light. This raw light-field image can be obtained by using the so called *Light-Field* cameras.

2. Obtaining EPI: After creating this set of perspective images, it is necessary to apply a reorganization of the input data in order to obtain an EPI image. The EPI image is a parametrization of a light-field, which consists slicing the stack of perspective images which represents it, as shown in Figure 36. This slice is a collection of straight lines, and the slopes of each line corresponds to the depth value of the point in the light-field image. If a light-field is a (x, y, u, v) parametrization, (x, y) being pixel positions and (u, v) being viewpoint position, then the EPI is a collection of (x, u) and (y, v) slices of the light-field. Parameters v, b, φ relate to each other as $b = v.tan(\varphi)$, where b is simply defined in [11] as a camera parameter, φ is the angle of each line in the (x, u) slice , and v maps to a disparity value of a pixel at point x . The EPI stack of 2D images are therefore extremely useful since they provide reliable information on the depth of each point the 3D world, which may lead to reductions of the dimensionality of a light-field.

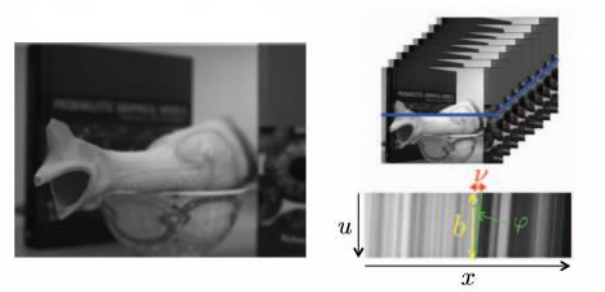

Figure 36: Light-field data representation through EPI. Left: Light-Field Image. Upper Right: Stack of perspective images. Lower Right: An EPI slice [11].

3. Applying Ray-Gaussian Kernel: Next, the scale-depth space is built by applying the Ray-Gaussian Kernel to the light-field slices (for more information on scale space and its purpose check Section 2.1), this kernel is defined as in (3. 1), and is heavily based on the Gaussian kernel used for scale-space construction in 2D images. The difference between the two lies in the term inside the exponential. While in the Gaussian kernel this corresponds to the mean value of a function, in the Ray-Gaussian kernel it corresponds to a ridge, with slant equal to $tan(\varphi)$. Figure 37 illustrates this kernel. While it will not be shown here, the scale-invariance of the Ray-Gaussian kernel was theoretically proved in [11] . The light-field scale and depth space, or LISAD is obtained by convolving a light-field slice with the Ray-Gaussian Kernel for several values of σ and φ, σ corresponds to scale , while φ corresponds to the angle of a line in the EPI. Since each line of the EPI corresponds to a depth value in the scene, applying a kernel which both scale and depth are parameters allows the creation of a scale-depth space.

$$R_{\sigma,\varphi}(x, u) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x+utan\varphi)^2}{2\sigma^2}} \qquad (3.\ 1)$$

Figure 37: Ray-Gaussian Kernel for $\varphi = \pi/4$ and $\sigma = 6$ [11].

4. **Applying Edge Detector:** After obtaining the so called LISAD, an edge detector is applied to each of its layers. While blob detectors are generally more reliable, edges are important in visual representations such as light-fields. Although in [11] the authors do not specify which edge detector is used, a few edge detectors are listed in Section 2.1.

Performance Assessment

The method used to test this new key-point detector in [11] consists of a comparison to the key-point detector used by the SURF descriptor [2], both in terms of efficiency and computational complexity. As such, images from the Middlebury database [43] are used, these images are taken from different viewpoints equally spaced on a line, and are therefore ideal for evaluating key-point detection in the context of light-fields.

data	method	% of keypoints with $\lvert\Delta d\rvert < \epsilon$				time
		$\epsilon = 1$	$\epsilon = 2$	$\epsilon = 3$	$\epsilon = 4$	(sec)
Teddy	SURF	51.05	66.41	72.94	77.93	14.31
	Lisad (a)	66.61	74.54	78.54	81.09	8.57
	Lisad (b)	60.04	71.74	76.92	79.01	5.51
Connes	SURF	46.16	64.42	75.00	79.14	18.16
	Lisad (a)	63.57	71.87	78.17	83.05	8.91
	Lisad (b)	57.66	71.53	77.38	82.53	4.71

Table 4: Comparison of Lisad and SURF [11].

Where Lisad indicates the new light-field key-point detector. Lisad (a) indicates an angle sampling step with 1 pixel disparity, while Lisad (b) indicates 2 pixel disparity. Teddy and Connes are two distinct image sets in the Middlebury database. The efficiency of the detector is measured using the percentage of key-points with disparity error less than ε, where ε varies from 1 to 4 pixels. All tests were conducted on a Linux system using a 2.6Ghz Intel CPU, and 64GB of RAM.

3.2.2. Improving Distinctiveness of BRISK Features Using Depth Maps

Objectives and Technical Approach

BRISK [3] is a binary feature descriptor, which extracts distinctive features by pair-wise intensity comparisons around pre-smoothed key-points. BRISK was explained in detail in Section 2.4. In [44] a method to improve the distinctiveness of BRISK features by including depth information of the visual scene to build a new BRISK-like sampling pattern is presented. Although this method is applied for BRISK, it could be applied for any binary descriptor. The key idea is to apply the BRISK sampling distribution (see Figure 22) as if it is attached to the 3D surface associated to each image, and apply the new sampling points in the usual way, to create a binary descriptor. This adaptation of BRISK is useful for a number of reasons. One of the most important being that it allows the better capture of

perspective changes in scenes, since these typically represent 3D motion, which is better described using depth information. Since this method is meant for texture + depth content, it is a good match for the Multiview + Depth content described in the basic concepts of this chapter.

Architecture and Main Techniques

The architecture of this descriptor is similar to the BRISK feature descriptor. The difference lies in the local parametrization of the sampling pattern, that is, in the estimation of the sphere that will be used for the distribution of the sampling points. In Figure 38 the architecture of the descriptor is shown.

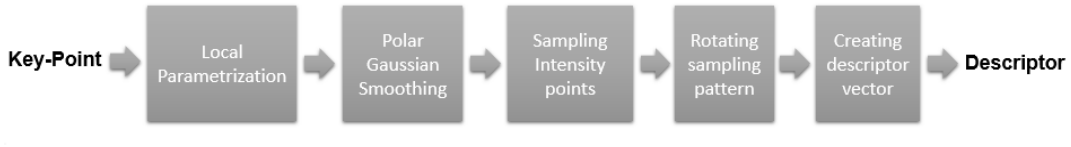

Figure 38: Architecture of depth-based BRISK descriptor.

1. **Local Parametrization:** The first step is to build the sphere in which the sampling points will be located, this sphere parametrization is important because it can describe the placement of sampling points in a 3D way. The sphere is composed of radial and angular components (see Figure 39). To compute the radial component $\rho(u,v)$ the authors use the following formulae [44]:

$$r(u,v) = \begin{pmatrix} 2u\tan\frac{w}{2} \\ 2v\frac{H}{W}\tan\frac{w}{2} \\ 1 \end{pmatrix} D(u,v) \qquad (3.\ 2)$$

$$R = \sigma D(u_0, v_0)\frac{2\tan\frac{w}{2}}{W} \qquad (3.\ 3)$$

Equation (3. 2) maps the position of each sampling point in the sphere, while (3. 3) determines the radius of the sphere. $D(u,v)$ is the depth value of the image at coordinates (u,v), H and W is its height and width in pixels, w is the horizontal angle of view of the camera and the depth sensor, σ is the key-point's scale, and (u_0, v_0) is the centre of the sampling pattern. Having computed this, the distances $\rho(u,v)$ from the sampling point centre (u_0, v_0) to the other pixels is determined through the fast-marching algorithm, which allows to compute the geodesic distance from a point on a surface to other points. Figure 39 shows a graphical interpretation of this radial component.

Regarding the angular component $\Phi(u,v)$, its computation is slightly less intuitive. Basically a contour of constant radius C is extracted from the sphere, the angles of the points in this contour are computed. With this operation it is possible to compute the angle of any other point in the sphere by selecting the angle the point in C which minimizes the angular distance between the two vectors from the key-point centre (for more detail on this step refer to [44]).

2. **Smoothing patch around key-points**: To smooth intensity values around a key-point, this method applies a modified version of the Gaussian Kernel that is often used in feature extraction methods. This proposed method works in polar coordinates, and takes into account the five layers of the BRISK sampling pattern, where the first layer is the key-point centre and each following layer

contains a set of points with constant radius (check Figure 22). The smoothing kernel therefore works differently for each layer and is defined as follows [44]:

$$K_{l,k} = \exp\left(-\frac{(\rho - r_l)^2}{2s_l^2} - \frac{\left(mod(\Phi - \frac{k-1}{n_l})\frac{n_l r_l}{4\pi}\right)^2}{2s_l^2} \right) \qquad (3.4)$$

Where r_l is the radius of layer , n_l is the number of points in that layer, r_l is the layer radius, and s_l is the layer scale. k is the sampling point number. The mod function converts the angle to radians. The parameters ρ and Φ are respectively the radial and angular components of a point. When this filter is applied at the selected sampling points, the smoothed sample values are obtained, these values will be used in the intensity comparisons later. In Figure 39 the smoothed sampling points are shown.

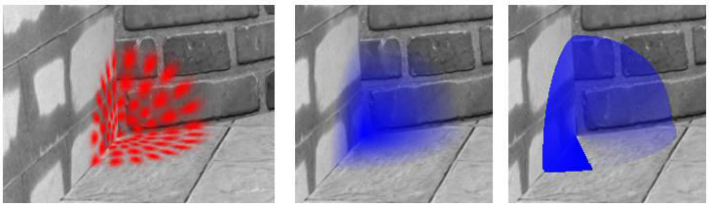

Figure 39: Left: New distribution of smoothed sampling points. Middle: Radial Component. Right: Angular Component [44].

3. Sampling Intensity Points: From this step forward the architecture is almost identical to the BRISK original architecture, first the new, depth-adapted sampling pattern that was estimated in the parametrization (step 1) with the smoothed sampling points computed in step 2 is applied.

4. Rotating Sampling Pattern: After, the sampling pattern is rotated according to the key-point orientation. The angular component $\Phi(u, v)$ is shifted according to the estimated orientation. The orientation is estimated according to the BRISK descriptor described in Section 2.4.2.2.

5. Creating Descriptor Vector: The final step is to compute the descriptor entries through pair-wise intensity comparisons. Just as in BRISK, only the short distance pairs are used to compute the descriptor itself. Short and long-distance pairs are pre-computed and are not adaptive to the patch that is being created as in the BRISK original creation.

Performance Assessment

This new descriptor is assessed in [44] by comparing it to other, 2D image descriptors such as SIFT [1], BRAND [45], and the 2D version of BRISK [3]. Each of these detectors was paired up with the BRISK descriptor, even though SIFT typically uses its own detector. The descriptors are tested on a texture + depth dataset which presents significant out-of-plane rotations (see [46] for examples). Figure 40 shows the ROC curves obtained on the entire dataset, for every descriptor that was tested.

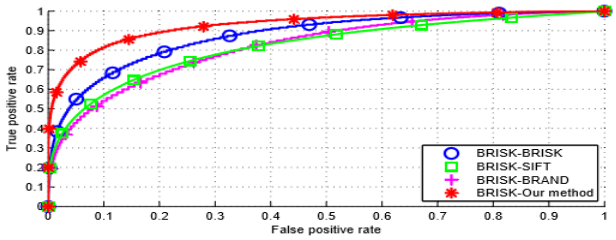

Figure 40: Receiver Operating Characteristics obtained on the entire dataset [44].

3.2.3. Scale Invariant Representation of Light-field Images

Objectives and Technical Approach

This method is presented in [10], with the purpose of providing a scale-invariant feature descriptor for light-field images. In this paper light-field images refer to images taken by the so called light-field camera. The method works by exploiting the fact that the slopes of EPI lines are proportional to the depth of the scene they represent. Using this information, the authors manage to extract a reliable scale invariant feature descriptor. To our knowledge this is so far, the only feature descriptor for images taken by a light-field (or Lytro-like) camera.

Architecture and Main Techniques

Figure 41 shows the architecture for this approach:

Figure 41: Architecture for Scale-Invariant representation of light-field images.

Since some of the steps performed by this descriptor are rather similar to those described in Section 3.2.1, these will not be described in detail here. The following are the steps computed to create a descriptor for light-field images.

1. **Creating Stack of Perspective Images**: Considering a raw light-field image as input, the first step is to convert this image into a set of 2D images, each one corresponding to a different perspective of the scene, just as in the architecture described in Section 3.2.1.

2. **Obtaining EPI:** The second step is to obtain Epipolar plane images representing the visual scene. This will provide a compact way to represent information on the depth of the scene. An example EPI is shown in Figure 36.

3. **Exponential Radon Transform:** Lines are the dominant objects in EPI, therefore it is useful to present EPI planes in a new space where detection of lines is easier. For this purpose, the Exponential Radon Transform is used, this is a variation of the Radon transform, which is often used for line detection [10]. In the exponential Radon transform, the image is transformed

according to parameters (λ, μ). The parameter λ is the compressed slope, and is obtained according to $\lambda = e^{-m}$, m being the slope value of an EPI line. While the second parameter $\mu = \lambda x + \lambda ln\lambda x$, (x, y) being the image 2D coordinates. The key property of this parametrization is that it is able to cancel uniform slope changes in the entire image. Considering λ_m as the compressed slope value λ corresponding to the slope value , adding a constant slope value Δ to every slope value m can be cancelled by [10]:

$$\lambda_{m+\Delta} = \lambda_m e^{-\Delta} \qquad (3.5)$$

4. Integration over λ-μ plane: Since the slope of each line is proportional to the depth of a certain point in the light-field, a simple way of obtaining a scale-invariant descriptor is to integrate over all slope values. By taking this into account the feature vector can be interpreted as an integration of the λ-μ plane over all possible slope values. The vector is constructed as follows [10]:

$$g(s) = \int E(\lambda, \mu)\delta(\mu - s\lambda)d\lambda d\mu \qquad (3.6)$$

Where E is the Exponential Radon Transform, and δ is the Dirac function used in signal processing. By using the Dirac function it is possible to obtain the integral as a finite set of values, which allows to create a fixed sized descriptor. Intuitively, this method can be justified by considering that changing the distance of a camera to a scene only changes the slope values of the EPI. Therefore, by integrating over the λ-μ plane it is possible to obtain information on every slope value, and consequently every scale value.

Performance Assessment

In [10] the presented descriptor is assessed through a dataset of light-field images created by the authors, and consisting of 240 light-field images. A single feature vector is extracted from each of the images, and compared to 2D descriptors DenseSIFT and HOG in terms of efficiency. The descriptor is assessed by measuring the percentage of correct classifications (efficiency), by measuring average query time per frame (computational complexity), and by measuring the feature length (compactness).

	Our Approach	*DenseSIFT*	*HOG*
Feature Length	**288**	12800	3100
Accuracy (% of Correct Classifications)	**88**	50	61
Average Query Time Per Frame (s)	0.06	**0.04**	1.8

Table 5: Results of the proposed method, compared to DenseSIFT and HOG [10].

3.2.4. Local Visual Features Extraction from Texture + Depth Content

Objectives and Technical Approach

The last solution presented is a feature extraction method that takes into account the depth of the visual scene to adapt a feature descriptor to the 3D visual representation, hence optimizing its performance. The method [46] provides better resilience to view-point changes and works by estimating local approximating planes to object surfaces to obtain slant-invariant texture patches, i.e. texture patches that are robust to out-of-plane rotations in the scene. Another relevant fact is that this

method is applied between feature detection and extraction, and due to this fact it can be used as a complement to various 2D feature descriptors.

Architecture and Main Techniques

Figure 42 illustrates the architecture of this method:

Figure 42: Architecture of feature texture + depth key-point processing.

1. Estimating Local Approximating planes: The first step is to obtain the normal vector to the texture surface at each key-point. Thus, it is necessary to exploit depth information of the visual scene. So assuming $d(i,j)$ as a depth value at pixel (i,j), (i_0,j_0) the key-point coordinates, and S the key-point area, the depth map in the region corresponding to the key-point area can be approximated with the following model: $f_{A,B,C}(x,y) = A(x-i_0) + B(y-j_0) + C$. The parameters A,B,C are computed by minimizing the average fitting error [46]:

$$F(A,B,C) = \sum_{(i,j)} \left| f_{A,B,C}(i,j) - d(i,j) \right|^2 \qquad (3.7)$$

2. Key-point Filtering: Using the information in the previous step it is possible to filter some key-points based on a few assumptions: 1) Since minimizing the fitting error is essential to estimate local approximating planes, key-points with a large fitting error are likely to have been poorly estimated, and therefore are filtered out; 2) key-points with large slant angles, i.e. the angle between the normal vector to the texture surface and the optical axis of the camera are unreliable since they might create sampling artefacts during the normalization phase. This angle is calculated as follows [46]:

$$\Theta = arctan\sqrt{A^2 + B^2} \qquad (3.8)$$

Therefore, key-points with a slant angle above the threshold $\Theta = 80°$ are removed.

3. Local Surface Sampling: After, it is possible to adapt the key-points to the 3D surface of the scene they represent. To do this a regular sampling grid window is built on the approximating plane $Ax + By - z + C = 0$ of each key-point. The key-point itself (i_0,j_0) is the centre of the window, and the window size is a function of the key-point scale and the slant angle. This is shown graphically in Figure 43.

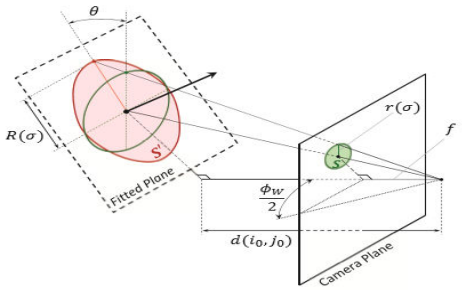

Figure 43: Estimation of sampling window size in the local approximating plane [46].

Where σ is the key-point scale and Φ_w is the horizontal angle of view of the camera, and $R(\sigma)$ is the size of the sampling window as a function of the key-point scale.

Performance Assessment

This approach is compared to the SIFT descriptor [1], as well as to the iterative normalization, suggested in [47]. The image sequences used for assessment are presented in [48] and are meant to test robustness to rotation. Table 6 shows a comparison of these three approaches in terms of matching accuracy.

Rotation angle, °	*Fish*			*Graffiti*		
	Raw	3D	Aff.	Raw	3D	Aff.
5	77.9	**78.8**	75.0	81.8	79.9	**80.5**
10	65.5	**70.9**	64.2	74.3	70.1	**73.8**
20	50.0	**56.2**	52.0	60.3	49.6	**60.2**
30	37.8	**52.1**	44.2	47.8	39.1	**53.0**
45	32.4	**46.4**	37.7	37.4	34.2	**47.6**
60	31.1	**44.0**	39.1	31.0	27.7	**43.2**
90	28.8	**36.8**	28.5	33.1	31.6	**41.6**
120	9.8	**16.3**	12.6	32.8	31.4	**37.2**
C_{max}	433	360	606	643	635	859
C_{min}	52	62	94	243	220	397

Table 6: Evaluation of the proposed descriptor to SIFT and the iterative normalization approach [46] .

Where Fish and Graffiti indicate two separate image sequences, while Raw, 3D, and Aff indicate the SIFT descriptor, the proposed approach, and the affine normalization, respectively. The table entries indicate matching accuracy, while C_{max} and C_{min} are the maximum and minimum number of correct matches for each descriptor – image set combination. For sequences with more complex geometry, i.e containing smooth convex surfaces, such as Fish, the proposed solution produces the best results, however for sequences with simpler geometry and more detailed texture such as Graffiti the proposed solution is outperformed, the authors justify this by the fact that small or low contrast texture details often produce false matches in their solution.

Chapter 4

4. Light-Field Dataset: Description and Retrieval Performance

To evaluate any feature detection or extraction scheme a suitable dataset is necessary. For this purpose, this chapter briefly reviews a few, already existing, light-field datasets, and proposes a new one, which addresses the main flaws and shortcomings of the already available light-field datasets, the main one being that there are no appropriate lenslet light-field datasets available for image retrieval, and the availability of this type of dataset is essential to understand and measure the benefits of using this emerging representation format. Another major goal of this chapter is to quantify the performance improvements obtained when lenslet light-field dataset is used for image retrieval, while still using classical 2D descriptors. Therefore, this chapter also evaluates the performance of standard 2D feature extraction methods on this dataset, and proposes a few novel feature matching frameworks that are adapted to the lenslet light-field format, i.e., light-fields captured by handheld light-field cameras such as the Lytro Illum.

4.1. Image Retrieval Datasets: Brief Review

To develop novel and efficient visual descriptors, it is critical to evaluate and compare their performance with the same data, and thus representative and meaningful visual datasets are needed. In the past, several datasets were proposed and are publicly available, each one containing different types of visual content (e.g. everyday objects, buildings) and using different visual representation formats. These datasets were designed for specific tasks, notably image/object retrieval, scene classification, and object detection and localization. In this section, the most relevant available datasets targeting image retrieval are presented. While there are many with 2D images, a few light-field datasets are already available. Table 7 lists some of the most relevant 2D image retrieval datasets.

Dataset Name	Categories	Images per Category	Format Type
CTurin180	180	8 + 1 video	2D images + video
Oxford Buildings	11	Varying quantity	2D images
Zurich Buildings	201	5	2D images
Stanford Mobile Visual Search	Varying quantity	Varying quantity	2D images
San Francisco Landmarks	125K	1	Panoramas

Table 7: List of retrieval datasets

The CTurin180 dataset [49] was created by Telecom Italia and contains sets of images and video sequences of landmarks from the city of Turin, Italy. The dataset was captured using 4 different cameras, an IPhone camera, a Samsung Galaxy S camera, and two Canon camera models. Each category contains a landmark captured from different viewpoints, as well as a rotated image of the landmark. The Oxford building dataset [50] was compiled by the Visual Geometry Group at the Univ. of Oxford and is a set of Flickr images containing landmarks from the city of Oxford, England. Each category contains a landmark captured from several different viewpoints and proximity levels, as well as detail shots of each landmark; for each category, 5 possible query images were established. This dataset also contains 100k distractor images from Flickr. The Zurich Building Image dataset [51], created by the Computer Vision Lab at ETH Zurich, contains images of 201 buildings in the city of Zurich, Switzerland. Each category contains a building captured at slightly different viewpoints, and occasionally with slight occlusion. A few categories in this dataset also contain rotated images of the buildings. The dataset also includes 115 separate query images. This dataset was captured using a Panasonic digital camera, as well as a Pentax camera. The Stanford Mobile Visual Search dataset [52], compiled by the Multimedia Systems Group at Stanford University, contains different sets of images taken by camera phones (e.g. books, CDs, paintings). Within each set, different products are separated into different categories with each category contains image transformations such as varying lighting conditions, foreground and background clutter, and different viewpoints. The San Francisco Landmark dataset *[53]*, also compiled at Stanford University, is a set of panoramas from buildings in the city of San Francisco, California, USA, captured using a vehicle mounted Ladybug 3 panoramic camera, as well as a high-definition Prosilica Camera. The dataset also includes a set of 803 query images captured using different consumer camera phones, with challenging viewing conditions such as clutter, poor lighting, and viewpoint changes.

Regarding light-field retrieval datasets, while a few do exist, most are not lenslet light-fields. Those publicly available are not large in size or don't provide a wide enough variety of image transformations. The EPFL buildings dataset [10] contains light-fields captured by a Lytro camera, but it is not publicly available. The images portrait a series of buildings located in the EPFL campus, in Lausanne, Switzerland. Other light-field datasets include the LCAV-31 [54], which is tailored for image classification rather than retrieval and was captured using a first generation Lytro camera; light-fields of 31 object categories are available. The categories consist of common household objects, e.g. clocks lamps, staplers. Considering the limited available light-field retrieval datasets, it is rather important to make publicly available a novel light-field dataset targeting image retrieval. This dataset should fill the 'empty space' by providing a wide variety of image transformations, which are not

always available in image retrieval datasets, thus allowing to make solid performance assessment in this area.

4.2. Lisbon Landmark Light-Field Dataset

This section introduces the Lisbon landmark light-field dataset. To be meaningful and useful for retrieval tasks. The choice of a landmark dataset to be used throughout this thesis was inspired both by the large amount of available landmark datasets for 2D images, and by the lack of available large landmark datasets currently available for light-fields. Although initially a second light-field image retrieval dataset was also captured, composed of different flower categories, some initial tests quickly proved that flowers were unideal for the testing of feature detection and description methods. The Lisbon light-field landmark dataset has been acquired under previously designed conditions defined in the following.

4.2.1. Dataset Description

The dataset proposed in this chapter consists in 25 landmarks from the city of Lisbon, Portugal. Each landmark was captured using a (hand held) Lytro Illum camera, which produces a raw light-field representation with resolution 225×623×484. After appropriate processing and rendering, it is possible to obtain a 15×15 array of views or so-called sub-aperture images corresponding to slightly different perspective positions, each with 623×424 pixels. For each landmark, a series of image variations was captured, notably:

- **Main view:** The landmark is captured from a frontal position, clearly visible, and at medium range.
- **Viewpoint Changes:** The landmark is still clearly visible, but is now captured from a different viewpoint than the main view; the difference in viewpoints may be more or less intense, notably including some extreme viewpoint angles, where the same features are less likely to be detected.
- **In-plane rotations:** Shots where the landmark is rotated with some level of in-plane rotation.
- **Varying distance:** The landmark is captured at a distance different from the main view, notably including shots where the landmark is very close or very far away.
- **Details:** Shots showing specific details for a given landmark, such as doors, windows, and sculptures in the facade.
- **Partial visibility:** Shots where only part of the landmark is shown, e.g. due to either occluded foreground, or actual camera position.

Figure 44 shows examples of the light-field central rendered view for each acquisition condition above. Although the examples shown in this figure were taken under ideal weather conditions and at the ideal time of day, it is important to note that some of the landmarks portrayed in this dataset were captured under unideal conditions, such as clouded skies. Making this dataset also adapted to varying weather conditions.

Figure 44: Rendered central view examples for various acquisition conditions, for two separate landmarks. From left to right and top to bottom: Central view, viewpoint change, in-plane rotation, long distance shot, detail shot, partial visibility shot.

4.2.2. Controlled versus Full Dataset

The variety of shot variations included in the proposed dataset makes it a very challenging dataset for image retrieval, and thus rather difficult for image retrieval algorithms. To consider more common acquisition conditions, the light-field dataset has been organized into two layers: a first layer dataset corresponding to shots taken under more Controlled conditions and a second layer including the Full dataset; naturally, the Controlled dataset is smaller than the Full dataset. The main objective of the Controlled dataset is to compare the performance of different 2D descriptors for more constrained acquisition conditions since some of the shot variations are too challenging. In summary, the Full dataset includes all the light-field images while the Controlled dataset excludes the most "difficult" shot 'transformations', e.g. drastic scale changes (relatively to main view shots), detail shots, and partial visibility shots; also, the most extreme viewpoint changes as well as in-plane rotation shots where the rotation angle is above 30 degrees (approximately) were excluded.

While the Full dataset includes around 1230 light-fields. This new, Controlled dataset contains around 450 images, hence the Full and Controlled datasets average 50 and 18 images per category, respectively. Table 8 displays the type and amount of image transformations in each dataset, note that these are round numbers and the actual number of each type of transformation might slightly vary from category to category. Also, as displayed in lower portion of Figure 44, these transformations are not mutually exclusive, i.e. a varying distance shot may also include some viewpoint change. Both the Controlled, and Full dataset will be publicly available.

Transformation	Full Dataset	Controlled Dataset
Main View	5	5
Viewpoint Changes	15 - 20	5-10
In-Plane Rotations	5	3-5
Varying Distances	15 - 20	-
Details	5	-
Partial Visibility	5	-
Average Number of Images per Category	50	18
Total Number of Images per Dataset	1230	448

Table 8: Amount of image transformations per category and dataset

4.3. Retrieval Performance Evaluation

Using the proposed Lisbon light-field dataset, it is possible to study the performance of well-known 2D descriptors. However, to exploit this novel light-field representation format a suitable retrieval evaluation framework is necessary.

4.3.1. Light-Field Evaluation Framework

The architecture of the light-field retrieval framework is shown in Figure 45.

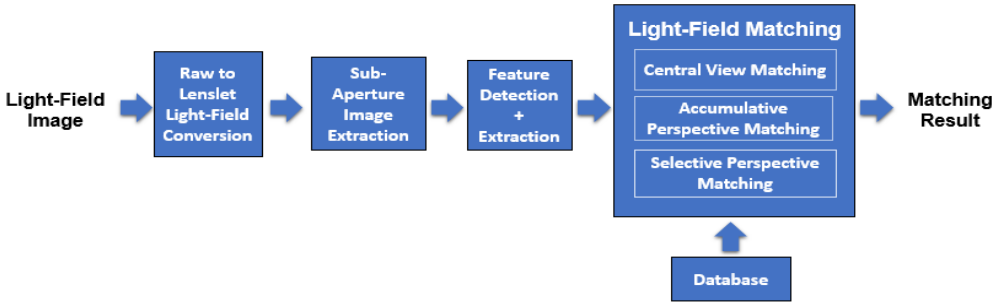

Figure 45: Light-Field processing and matching framework

This framework includes the following steps:

1. Raw to Lenslet Light-Field Conversion: The first step is to process the raw light-field representation format provided by the Lytro Illum camera that was used in the acquisition phase. In this step, a 4D light-field, i.e. a 4D dimensional array with two ray directions and two spatial indices of pixel RGB data, is obtained. In this case, demosaicing, devignetting, transforming and slicing, and finally color correction is performed. The pre-processing applied over the raw light-field data is done using the Light-field Toolbox developed by D. Dansereau [55].

2. Sub-Aperture Image Extraction: The 4D array obtained in the previous step has the pixel values in RGB for all the perspectives captured with the lenslet camera. Thus, by rearranging this data it is possible to obtain a set of 225 sub-aperture images, each one corresponding to the perspective view captured considering the angular resolution of the camera, in this case a 15×15 grid of images, thus supporting horizontal and vertical parallax.

3. Feature Detection + Extraction: In this step, from each selected sub-aperture image, a set of key-points locations are identified using a feature detector; the area surrounding each key-point is then used as input to a feature extractor to obtain the final descriptor. Naturally, performing feature detection for all the sub-aperture images is heavy and often does not improve the performance, e.g. disparity is rather small among adjacent images in a lenslet camera. Thus, the amount and position of sub-aperture images in the 4D array for which feature detection and extraction will be performed can be defined *a-priori* (as defined in Section 4.3.2). The SIFT, SURF, ORB feature detectors and extractors were used, as well as the BRIEF descriptor using CenSurE STAR [16] as detector. These feature detectors and extractors have different trade-offs in terms of complexity and performance. The OpenCV 2.4.9 [56] implementation was chosen for all the feature detection and extraction methods selected. After feature detection, all key-points are sorted according to their reliability, in this case using the Hessian response as sorting criterion. Only the 250 key-points with the highest response were selected for each sub-aperture image, while the remaining were simply discarded. This number of key-points provides a good trade-off between a high retrieval performance and a relatively compact description.

4. Light-Field Matching: The matching between the query light-field image and all the database light-fields is performed to rank them according to a meaningful similarity criterion. In this module, a

similarity score (LF_C) between the query light-field and each database light-field is computed. This score is computed by matching one or more pairs of sub-aperture images for which descriptors were computed, and obtaining a sub-aperture similarity score (SA_C) for each sub-aperture pair. To compute the LF_C score, it is also necessary to select the sub-aperture image pairs for which SA_C is computed. Therefore, three proposed alternative solutions are defined in Section 4.3.2. To compute the SA_C score, it is necessary to perform matching between descriptors from two sub-aperture images, obtained from a query light-field and a database light-field. The following procedure is used:

a) Brute force matcher (BFM): correspondences between each descriptor from the sub-aperture query image and the descriptor of some database image are found by assigning each query descriptor to its nearest neighbor. The Hamming and the L2 distances are used for binary and real-value descriptors, respectively.

b) Ratio test filtering (RTF): Each correspondence provided by the BFM is then filtered through a simple but yet effective ratio test [1], to eliminate potential "wrong" matches. This test compares the ratio of distances between the two top correspondences for a given key-point and rejects the top correspondence if the ratio is above a threshold of 0.75.

c) Symmetric filtering (SF): After, symmetric matching is performed, i.e. each pair of sub-aperture images is matched, but this time descriptors of the database sub-aperture image are matched to the descriptors of the query sub-aperture image. This results into a second set of correspondences, which are filtered with the same ratio test as before. Then, the two correspondence sets are compared side-by-side. Considering that (*m,n*) are the descriptor indexes for each sub-aperture image, if a matched pair (*n,m*) in the first set has a corresponding pair (*m,n*) in the second set, then (*n,m*) is accepted as a match; otherwise, the match is discarded.

The sub-aperture similarity score SA_C is just the number of correspondences left (i.e. inliers) after the BFM, RTF and SF.

4.3.2. Light-Field Similarity Matching Score

In this work, different light-field similarity matching scores LF_C are proposed to exploit the richer light-field image and the powerful local 2D descriptors, such as SIFT and SURF. The key difference between each score lies in how the sets of sub-aperture images corresponding to each light-field are matched. The following three solutions are proposed:

- **Central View Matching (CVM)**: In this solution, only the central sub-aperture image, i.e. the (8,8) position in the 15×15 grid, is extracted. This is the poorest form of light-field image retrieval, since the light-field image is represented just as a 2D image. This sub-aperture image, which corresponds to the central perspective, can be matched to all other central sub-aperture images in the light-field database. In this case, the LF_C score is equal to the SA_C score, i.e. the best match for

this method is simply the light-field image whose central sub-aperture image has the largest SA_C with the query. This solution is very simple and takes no advantage of the extra light-field information with respect to standard images. However, it can be used as benchmark for better solutions.

- **Accumulative Perspective Matching (ACM)**: This solution selects 5 different light-field sub-aperture images for the matching process, in this case, the central (8,8), an upper (8,2), a lower (8,14), a right side (2,8), and a left side (14,8) perspective. These sub-aperture images were selected to cover the widest possible range of perspectives in the light-field image while maintaining a limited amount of extra information. The sub-aperture images in the very top, very bottom, left most, and right most positions were not picked because they are often too dark (or with optical aberrations). In this solution, for each query, these 5 sub-aperture images are matched to each *corresponding* sub-aperture image (i.e. in the same position) of each light-field in the dataset. The SA_C score across all 5 sub-aperture images is then added to obtain LF_C and the light-field database images ranked according to LF_C. This solution exploits the light-field representation format, although not fully. The key idea is to reinforce true matches between sub-aperture images by not matching a single perspective (as in the previous method) but rather several perspectives.

- **Selective Perspective Matching (SPM)***:* This solution is similar to the previous one since the same 5 sub-aperture images are considered. However, instead of matching sub-aperture images from the query and light-field database in the same corresponding position, this solution follows a different approach. Each selected sub-aperture image from the query light-field is matched to all selected sub-apertures in the light-field database image and thus all combinations are possible. The matching with the largest SA_C, which is not necessarily for two sub-aperture images in the same position, is retained. This process is repeated for all 5 sub-aperture images from the light-field query image. The largest SA_C scores across all query sub-aperture images are then added to obtain LF_C, and the light-field database images are ranked according to it. This solution was designed to adjust the matching process for a more unstructured scenario where two light-field images can have any arbitrary real scene position and thus sub-aperture image may best match in non-corresponding positions.

4.3.3. Matching Experimental Results

The query light-field images were randomly selected within the dataset. For the Full and Controlled datasets, 3 query light-field images were selected for each of the 25 landmarks, adding to a total of 75 query light-fields for each dataset. To evaluate the retrieval precision, two metrics were selected: the Mean Average Precision (MAP), and the top level precision (PA1). Table 9 show the experimental results obtained for these two performance metrics. The results obtained using the Controlled and Full datasets are in the Controlled and Full columns, respectively. The experimental results (in %) are shown for the CVM, APM, and SPM matching criteria presented above.

MAP	Full			Controlled		
	CVM	APM	SPM	CVM	APM	SPM
SIFT	31.4	37.2	**38.4**	56.6	63.1	**64.6**
SURF	26.1	33.5	**34.7**	51.3	60.8	**62.1**
ORB	18.7	26.8	**28.1**	39.5	50.9	**53.7**
BRIEF	12.4	24.2	**24.4**	30.6	37.7	**37.8**

PA1	Full			Controlled		
	CVM	APM	SPM	CVM	APM	SPM
SIFT	**91.8**	90.4	90.4	91.3	**94.2**	**94.2**
SURF	82.2	**91.8**	90.4	91.3	**92.7**	**92.7**
ORB	79.4	80.8	**86.3**	86.9	89.9	**91.3**
BRIEF	57.4	81.2	**87**	75.4	79.7	**82.6**

Table 9: Left: Mean average precision experimental results.
Right: Top level precision experimental results

By analysing Table 9 it is clear that the APM brings significant MAP improvements with respect to CVM regardless of which descriptor is used. The MAP performance using the controlled dataset improves up to 11%. With SPM, the MAP performance can be improved up to 3% with respect to ACM. The MAP performance gains provided by SPM would increase if a camera system with a wider range of sub-aperture images would be used, since the Lytro Illum microlens array provides a set of sub-aperture images with rather limited disparity variations between them.

Regarding the MAP performance of each tested descriptor, SIFT, SURF, ORB and BRIEF, the performance differences between them are as expected. The real-valued descriptors perform considerably better, with SIFT having a slight edge over SURF. Despite this, SURF is the real-valued descriptor benefiting the most from the light-field matching solutions, increasing by a total of 11% using the controlled dataset, and 9% using the full dataset (SPM with respect to CVM); SIFT improves only 8% using the controlled dataset, and 7% using the full dataset. Regarding the binary descriptors, ORB performed considerably better than BRIEF, as expected. While BRIEF was the descriptor whose performance increased the most using the full dataset (12%), ORB was ultimately the descriptor benefiting the most, since its performance under the Controlled dataset increased 14%.

Regarding the top-level precision performance, real-valued descriptors with light-field matching (APM/SPM vs CVM) barely improve and sometimes minimal losses are observed, e.g. SIFT for the controlled dataset. However, the PA1 values for real-valued descriptors are already high even when using single view matching (CVM) and thus it is more challenging to significantly improve their performance. On the other hand, the binary descriptors tend to show more stable improvements, with BRIEF standing out by showing an improvement of 30% using the full dataset.

Lastly, it is important to note the large performance discrepancies when using the Full and the Controlled datasets. This validates the motivation to create the two datasets and confirms that the inclusion and exclusion of certain image transformations in a retrieval dataset largely affect the overall retrieval performance.

4.4. Final Remarks

Aside from presenting a new lenslet light-field retrieval dataset that is highly useful and to this date was not available, this chapter has served has proof that using extra information provided by light-fields can considerably increase performance of feature detection and extraction methods. One can also predict that the gains that the proposed alternative matching methods are proportional to the disparity that a set of sub-aperture images from a light-field presents.

Chapter 5

5. Light-Field Key-Point Detection

This chapter proposes a novel light-field key-point detector that can be applied to the lenslet dataset that was acquired in Lisbon and described in the previous chapter. However, the main concepts and framework can ideally be extended to other light-field formats, such as a dense array of cameras. The developed solution exploits the so-called *Epipolar Plane Image* (or EPI) which is created using a set of perspectives rendered from a light-field.

First, this chapter introduces the concept of Epipolar geometry, and the Epipolar Plane Image, and explains why it is relevant for the case of light-fields. After this introduction, the framework proposed for the key-point detector is described. Next the main tools used for this detector are also described in detail. After that, the evaluation framework used to assess the performance of this solution is presented and explained. Following this, a few important implementation options are explained and justified. Lastly, the performance of this solution is compared to the 2D SIFT key-point detector under different image transformations. To clearly make the distinction between key-points, which are detected in a single sub-aperture 2D image, and the output of our detector, which is the result of processing multiple sub-aperture images, the output of our novel detector is referred as “key-locations”.

5.1. Epipolar Geometry

Epipolar geometry is the geometry usually employed in visual representation sensors and formats which go beyond the use of a single 2D sensor. In chapter 3, several of these representation formats were described. The different perspectives of a visual scene acquired or represented with these formats enable the extraction of information regarding the scene which would not have been available using a single sensor, namely depth information. Epipolar geometry is heavily inspired by stereopsis, i.e. the perception of depth and 3D structure from the visual stimulus provided to both eyes. Naturally, the objective of the EPI is to represent the same information using the information captured with visual sensors as the human brain does using the eyes. By capturing the light in the scene using an array of cameras or with a lenslet based sensor, this means capturing also the light direction, it is possible to estimate the depth of objects in this scene. One way to do this is to “slice” the various acquired views at the same line; putting together all these lines for the multiple views enables the computation of the displacement of the same point between views, this means the disparity, and consequently the depth

of the object at those line coordinates. This slice is called the Epipolar Plane Image (EPI); see an example in Figure 46.

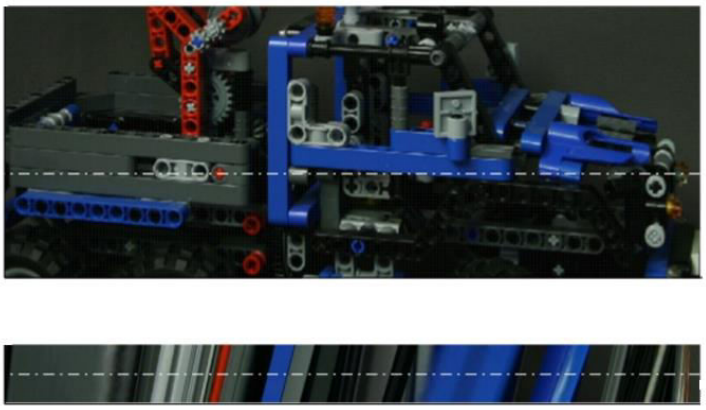

Figure 46: Example of Epipolar Plane Image [57].

The spatial characteristics of an EPI depend directly on the type and displacement of the sensors which were used to acquire the views. Since a lenslet light-field is acquired by an array of micro-lenses, it is possible to obtain from this representation sets of EPIs (for the various lines) which provide rich information on the visual scene. The EPI representation is, therefore, a key part of our key-location detector, which is described in the following section.

5.2. Light-field Key-Location Detection Framework

Figure 47 displays the framework used for our light-field key-location detection solution.

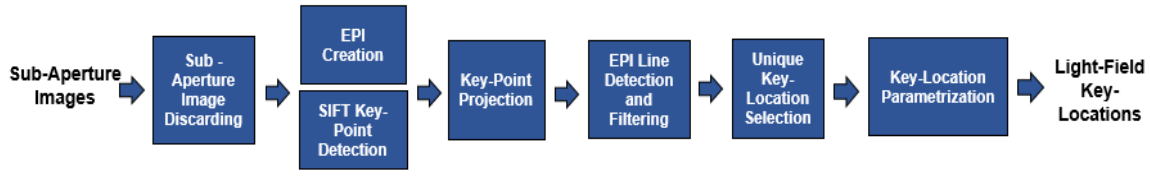

Figure 47: Light-field key-location detection framework

Each of the seven steps will now be explained:

1. **Sub-Aperture Image Discarding:** This module receives as input a 2D array of sub-aperture images corresponding to a full lenslet light-field. First, every sub-aperture image is converted to grayscale before any processing. The mean luminance value L_C of the central sub-aperture image is computed. Then, any sub-aperture image with a mean luminance value below $0.35 \times L_C$ is discarded. This is performed to discard images that are too dark to provide any useful information in terms of feature detection, which are often the images corresponding to the corners of the light-field sub-aperture array. Although the 2D sub-aperture images which are later selected for this light-field detector are typically not near the corners of the array, and therefore not needed to be discarded, this step can be especially useful if the detector is eventually extended to select other

sub-aperture images, or when due to the camera characteristics or other illumination changes some of the selected sub-aperture images are too dark to be useful for key-point detection. Figure 48 shows a comparison of the up-most sub-aperture image in a light-field, compared to its central sub-aperture image, by analysing these two images one can see the difference in terms of subjective quality between the two images, for example in the blurred branches of the small tree, closer to the camera.

Figure 48: Left: Central sub-aperture image. Right: Up-most sub-aperture image.

2. EPI Creation: In this step, two EPI 3D arrays (or EPI cubes) are created from the arrays of sub-aperture images, using only the sub-aperture images from the central row and the central column. This means that one vertical set, and one horizontal set of sub-aperture images are used; these two sets both cross the sub-aperture image array at its center and are used to create the vertical and horizontal EPIs. As explained in Section 5.1, an EPI is a two-dimensional slice of a light-field obtained by sampling its sub-aperture images at a specific coordinate. The size of each EPI depends on the number of sub-aperture images in the processed light-field, and the dimensions of each individual sub-aperture image. In the case of the *Lytro Lenslet Light-fields*, both the horizontal and vertical directions have 15 sub-aperture images, where each sub-aperture image has a size of 623 by 434 pixels. Therefore, a central horizontal EPI should be of size 623 by 15 pixels, while a central vertical EPI should be of size 434 by 15 pixels. When all EPIs from an array of sub-aperture images are joined together, an EPI cube is created. Figure 49 illustrates the creation of a vertical EPI. In this case, a vertical array of sub-aperture images from a lenslet light-field is sliced at a specific *x* coordinate, and the pixel values extracted from each image are stacked on top of each other to form the EPI.

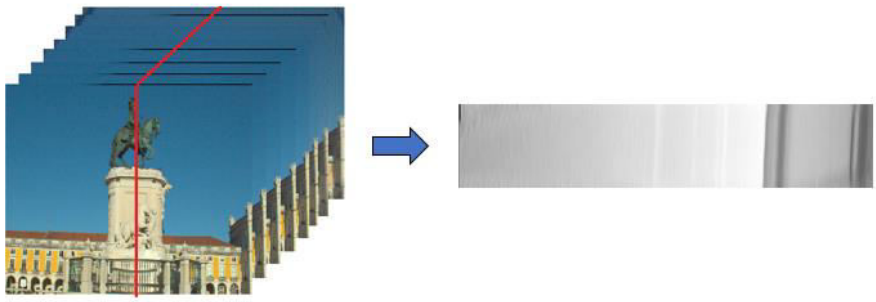

Figure 49: Slice for a stack of sub-aperture images used to create a vertical EPI. The EPI resolution is here increased since its original height is only 15 pixels.

3. SIFT Key-Point Detection: The SIFT feature detector is applied for all the sub-aperture images in the central horizontal and central vertical arrays. The 500 key-points with highest response for each sub-aperture image are kept, while the remaining key-points are discarded. The SIFT detector was chosen as a starting point since it is one of the most reliable and high-performance feature detectors; moreover, it estimates the size and orientation of key-points, which is useful both for feature detection and also description, which is another target of this Thesis.

4. Key-Point Projection: This step consists in the projection of the key-points obtained in the previous step to their corresponding EPI in the EPI cube. This is done by computing the coordinates of each key-point in the EPI cube from the corresponding coordinates in the array of sub-aperture images. This step is performed for the horizontal and vertical EPI cubes. For example, if a key-point is detected in the 2nd image of the central horizontal array (counting from left to right), at coordinates (100, 75), then the coordinates in the horizontal EPI cube are (100, 2) in the 75^{th} EPI of the horizontal EPI cube, (which has a total of 434 EPI). By performing this projection for every detected key-point, it is possible to later compute how many times each key-point was detected; this is important since the number of times the same key-point is detected in an EPI cube is a good indicator of the subjective importance of that key-point for the corresponding light-field. However, determining whether the same key-point is detected or not in both the horizontal and vertical EPI cubes is done in a later processing step. Figure 49 shows sets of key-points detected in the sub-aperture images projected to two different EPIs. Note that in this figure the resolution of the original EPI was increased 10 times in the vertical axis, since each EPI has a height of only 15 pixels. After the key-point projection process, it is possible to define a light-field key-location as a line formed by sets of key-points in different sub-aperture images. In the top EPI of Figure 49, it can be observed that a single key-location was detected, since key-points from the same visual feature are repeatedly detected in several sub-aperture images of the array, since they define a straight line. In the bottom EPI, four distinct key-locations are detected, since there are four distinct lines of detected key-points. One of these key-locations is composed of only a single detected key-point, this is not a “strong” key-location, and will be filtered out in the next step of our framework.

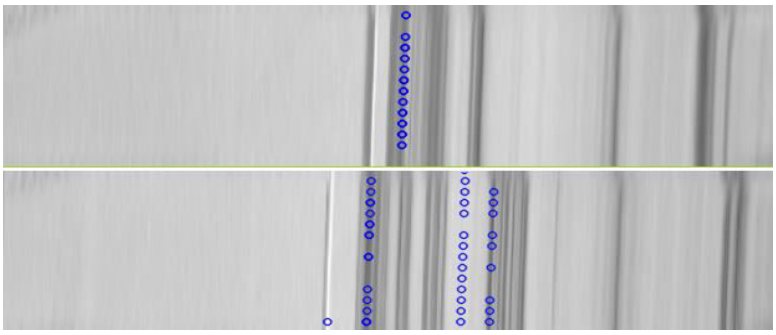

Figure 50: Key-points detected in sub-aperture images projected to an EPI.

5. EPI Line Detection and Filtering: This step analyses each EPI to identify any straight line that might be present; these straight lines are important since they represent a sequence of detected key-points at the same location, but in different sub-aperture images. This step is performed to determine the angle (and other parameters) of each key-location, i.e. the placement of the key-points belonging to each key-location along the vertical axis of each EPI. Each line is defined by a pair of values ρ and *Θ* (as in Equation 5.1), where *ρ* corresponds to the perpendicular distance from the origin of the image coordinate system to the line measured in pixels, and *Θ* is the angle formed by this perpendicular line and the horizontal axis as shown in Figure 50, note that our software uses an inverted coordinate system, hence the inverted vertical axis in Figure 50 .

$$\rho = x\cos\Theta + y\sin\Theta \qquad (5.1)$$

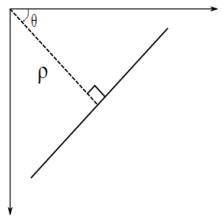

Figure 51: Straight Line parameter illustration [58].

To perform the straight lines detection, a suitable transform is necessary; in this case, the Hough Transform was selected [58]. This transform has proven effective for these tasks, but it requires some posterior filtering to effectively detect the relevant key-locations in the light-field. Therefore, after applying the Hough Transform, a well-defined sequence of filtering steps is performed. This process is described in Section 5.3.

6. Key-Location Parametrization: The final step in this framework is to compute the characterizing parameters for each retained light-field key-location and store them in a suitable data structure. This structure contains the following data:

- Information associated to the several SIFT key-points associated to each key-location, i.e. coordinates, size, rotation, response, and octave.

- Information associated to the key-location itself, i.e. its *ρ* and *Θ* values, as well as its score defined as the number of key-points it contains, and its depth d, which is associated to the slope

of the line measured by means of the Θ value for each key-location using the following expression:

$$d = \frac{1}{tan(90-\Theta)} \qquad (5.2)$$

7. Key-Location Selection: Since each key-location was either detected in the horizontal or vertical EPI cube, it is likely that different key-locations refer to the same exact coordinates in the light-field. Thus, the final step is to search for duplicate key-locations. This is done first by averaging the coordinates of the key-points inside each key-location, to obtain a pair of coordinates which are representative of that specific key-location. The average size of each key-location is also estimated by computing the average size of every key-point contained in the key-location. If the average coordinates of a key-location in the horizontal EPI cube fall within a 1 pixel window of the average coordinates of a key-location in the vertical EPI cube, and if the average sizes of both key-locations are within a 10 pixel range of each other, these are considered duplicates, and only one of the key-locations is retained, in our case the retained key-location is the one present in the horizontal array, although this decision is ultimately irrelevant. After this filtering process is concluded, the remaining key-locations are sorted according to some criteria. The different sorting methods and criteria are described in Section 5.4.5.

5.3. Main Tools

This section describes the technical details of some key tools mentioned in the walkthrough above, notably the Hough Transform and the following key-location filtering process.

5.3.1. The Hough Transform

The Hough Transform can detect lines in an image by creating a 2D grid which represents a space suitable for the detection of lines. The cells of this 2D grid correspond to the ρ and Θ parameters present in Equation 5.1 which can define any straight line. The size and number of cells in this grid directly depend on the precision of the ρ and Θ parameters, the better the precision of the ρ and Θ parameters, the greater the number of cells. Before applying this transform, it is necessary to convert the input image to the binary format, with only two possible pixel values, black and white. This is usually done using a Canny Edge detector [59]. However, in our case, the most effective solution is to simply create a binary image with the dimensions of the EPI, with a white value (here a '1') placed at the coordinates where each key-point was detected.

The Hough transform is an iterative procedure where, for every white point (key-point) in the binary image, Equation 5.1 extracts every possible ρ and Θ value present in the 2D grid, i.e. every possible straight line in the grid that passes through that point. Then the value inside the grid cells corresponding to those lines is incremented. Applying this process for all the white points, the grid cells corresponding to actual straight lines will have a much higher score, i.e. the number of times the cell was incremented should be higher than for other grid cells. Therefore, by thresholding the value inside each grid position, i.e. by setting a minimum value for the score of each grid position, it is

possible to identify the valid lines. Figure 51 shows a graphical illustration of this process; while this is just a 2D image example, it shows how the Hough Transform works.

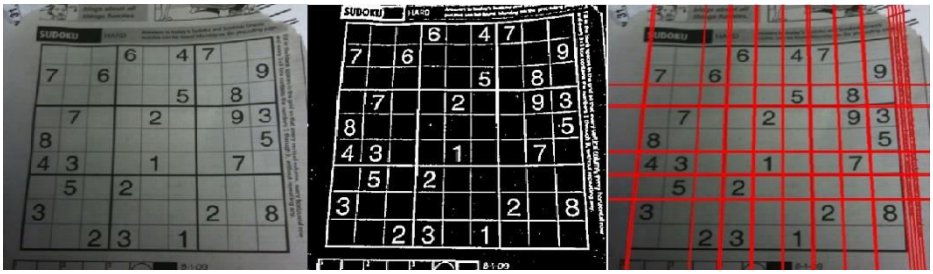

Figure 52: Left: Grey-scale image. Middle: Binary Image. Right: Detected lines over grey-scale image [59].

In this work, different Hough transform implementations were evaluated, to finally select the scikit-image [60] implementation mainly because it produces the best results (more details are described in the Section 5.4.3).

5.3.2. Post Hough Transform Line Filtering

To improve the performance of the light-field key-point detector the output of the Hough Transform is processed and some lines that were first detected are finally removed. The following filtering steps are performed:

1. **Filtering by score:** Since each detected line has an associated score, i.e. the number of key-points belonging to that line, the first filtering step is to discard lines with a small score, i.e. with just a few points. As this threshold is crucial for our key-point detector, several threshold values were tested, which results are shown in the next section.

2. **Filtering by Θ range:** The lines defined by the sets of key-points display a very limited range of Θ values; this happens because the slope of these lines is proportional to the disparity among sub-aperture images in a light-field, and this slope range is very limited for lenslet light-fields. Therefore, any detected line which slope does not lie within a maximum and minimum range should be discarded because it is very likely an 'outlier' line. In this case, any detected lines which do not meet the condition $-45° \leq \Theta \leq 45°$ are immediately discarded.

3. **Filtering by proximity:** The third filtering step is to identify sets of lines which are rather close to each other and thus very likely represent the same key-location. This effect occurs because key-points do not lie perfectly in a line and, due some imprecisions, there are small variations in their location among the different sub-aperture images (which correspond to different perspectives). This means that to obtain an accurate line representation it is necessary to use small grid cells and, therefore, multiple lines can be obtained in nearby locations. To solve this problem, the set of detected lines is analysed and only those which are at least 45° and 10 pixels apart from each other are retained, as lines within this range do probably correspond to the same key-location.

4. **Filtering by intersection checking*:*** The final filtering step is to analyse the remaining lines in each EPI, and check if any of them intersect each other. If two (or more) lines in the same EPI intersect each other, the line which contains the largest amount of key-points is retained, while the others are discarded. Typically, the line retained corresponds to the longest line, which means that the same feature was detected in a larger amount of views and, therefore, it is more stable. This step is performed to act as a complement to the previous step, to assure that no repeated lines are carried on to the output of the detector.

The bottom part of Figure 52 shows two detected lines obtained at the end of this step, i.e. after every necessary filtering step is applied. Note that while there are three possible key-locations (lines), only two are finally kept; this occurs because one of the possible key-locations does not have the necessary number of key-points to define a straight line, which in the case of the figure is six.

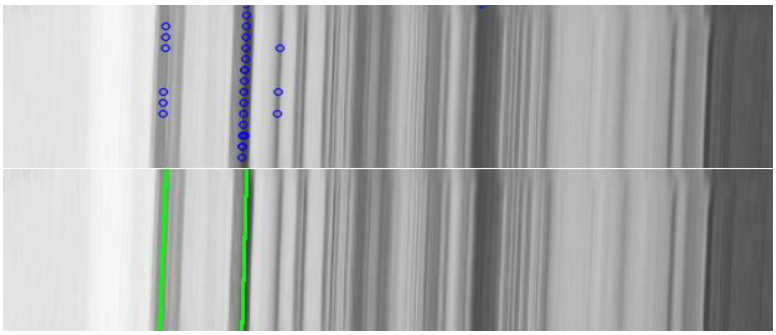

Figure 53: Top: Detected set of key-points. Bottom: Filtered Hough Transform lines.

5.4. Light-field Key-Location Detector Evaluation

To evaluate the impact of the various key-location detector options (such as the post-filtering) described above on the overall performance, a suitable evaluation framework is necessary. Within this framework, it should also be possible to compare the final light-field detector with other 2D key-point detectors. The evaluation framework proposed here is based on a few already available solutions, and its basic idea is to evaluate the robustness of our detector to several typical image transformations, most of which can occur in real scenarios.

5.4.1. Visual Transformations for Evaluation

Based on the available literature [61], some image transformations were selected to evaluate the light-field detector's robustness:

- **Viewpoint change:** A pair of light-fields portraying a visual scene from two slightly different perspectives, i.e. acquired from a different angle.
- **Scaling:** A pair of light-fields where the first is a zoomed-in version of the second.
- **Rotation:** A pair of light-fields where the first is a rotated version of the second.
- **Blur:** A pair of light-fields where the first is a blurred version of the second.

Apart from the blur transformation, which was performed by applying a Gaussian smoothing filter, every transformation is a pair of light-fields obtained from the Lisbon light-field landmark dataset, i.e. no synthetic transformations were applied to obtain one light-field from the other. Figure 53 shows examples of the rendered central views for each transformation pair. To evaluate the impact of some parameters on the light-field key-location detector, two pairs for each transformation type were used; however, for the final performance assessment, i.e. comparing the proposed light-field key-location detector with other key-point detectors, two additional pairs were added, making a total of four pairs per transformation.

Figure 54: Examples of image transformations. From left to right, and top to bottom: Viewpoint change, scaling, rotation, blurring.

5.4.2. Light-field Key-Location Detector Evaluation Framework

Our evaluation framework receives as input two sets of light-field key-locations corresponding to the two light-fields in a transformation pair. Each pair of light-field transformations is processed in a similar way to measure the performance of the light-field key-location detector. Naturally, the idea is to assess the number of key-locations that are similar/stable from one light-field to the other in the pair; when the same key-locations are obtained in both light-fields, they can be declared as stable. Figure 54 shows the evaluation framework.

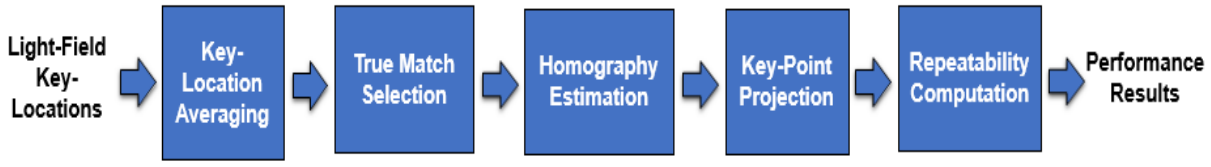

Figure 55: Light-field key-location detector evaluation framework.

Next, the walkthrough of the evaluation framework is presented:

1. **Key-Location Averaging:** As explained in Section 5.1, each key-location is defined as a data structure corresponding to a straight line defined by ρ and Θ, score and depth, as well as a set of key-points which are contained in that straight line. To compare the performance of this detector with a 2D key-point detector, it is necessary to evaluate them under the same conditions and, therefore, these key-locations need to be first converted to 2D key-points. To do this, some of the

attributes of the set of key-points associated to each key-location, i.e., size, angle, and response, are averaged to create a single representative key-point for the entire key-location; the spatial coordinates of this single key- point, i.e. *x,y,* are the coordinates of the key-point in the key-location which is closest to the central sub-aperture image. This new key-point should, in theory, be more robust, since it is obtained by performing detection over an array of images, instead of just one. As this framework works with two light-fields, this process has to be repeated for every detected key-location in both light-fields.

2. True Match Selection: To compute the true relationship between two images, two well defined sets of coordinates, one from each image in the pair, are necessary; these two sets of coordinates must correspond to the exact same feature in both images. So to make this evaluation framework as robust as possible, a suitable matching process is necessary. In this case, these sets were obtained by applying the matching process described in the light-field matching module of Section 4.3.1 for the central views of each light-field and selecting the 30 strongest matches. An additional verification to check that every match was indeed correct was performed manually, any matches that are not correct are discarded and not used for the following step. Figure 55 shows two sets of points to be used for the homography estimation.

3. Homography Estimation: The relationship between two images representing the same visual scene from different angles can be efficiently described by a homography. After having obtained the two sets of coordinates in the previous step, a homography [62] between the two images is estimated. Here, the homography is a 3×3 matrix representing how two planar surfaces are related, i.e. how the pixel coordinates in one image correspond to the pixel coordinates in another image assuming planar surfaces which is not always the case. Naturally, this step (and the next) is skipped in the case of the blur image transformations, since for these transformations the key-points should be placed at the exact same coordinates, and therefore no homography is needed to describe their “displacement” from one image to another.

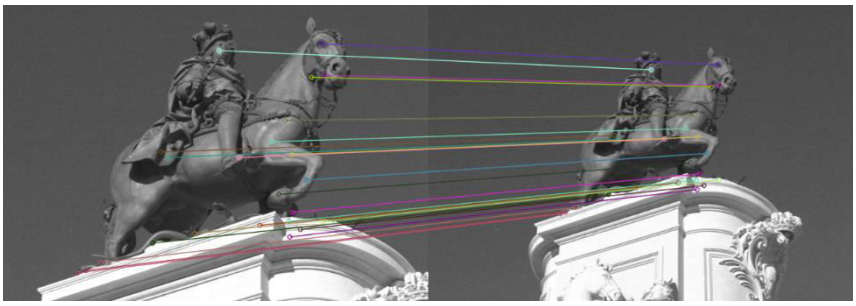

Figure 56: Key-point matches used for homograhy.

4. Key-Point Projection: This step uses the homography computed in the previous phase to project the key-points detected in one image to the other. This will enable the computation of the so-called *repeatability score*, which measures the robustness of the proposed light-field key-point detector to the selected image transformations. Figure 56 shows an example with the process of

key-location projection. Notice how the projected unique key-locations only cover the salient objects present in the two images.

Figure 57: Key-point projection example: Left: Key-points in the first image. Middle: Key-points detected in the second image. Right: Key-points detected in the first image, projected to the second image.

5. Repeatability Computation: The final step is to compute the repeatability of the key-points. Repeatability can be defined as whether the same key-point was detected in the two versions of the same visual scene. To compute this metric two solutions were considered:

- **Repeatability criterion 1 (R_{pos}):** The most intuitive approach is to create a small spatial window with some specific size (in this case 5x5 pixels) around each projected key-point, and check whether or not there is a key-point in the second image that falls within this window if this happens, a repeatability score is incremented. The total of this score for the full image is divided by the minimum value between the projected key-points from the first image, and the detected key-points in the second image, as follows:

$$R = \frac{score}{min(projected_{kp}, detected_{kp})} \quad (5.\ 3)$$

- **Repeatability criterion 2 (R_{all}):** Since there is no guarantee that the key-points that fall within the same window are actually the same key-points, previous approach can be improved to be more robust. Therefore, to perform the final assessment, a second repeatability criterion was adopted that takes into account the position, angle and scale of each key-point. This repeatability metric was defined in [63] as the ratio between the number of point-to-point correspondences and the minimum number of key-points detected for a given pair of images, this approach takes orientation and scale of key-points into account by defining an elliptical area around each key-point with size proportional to its size and rotates that area according to the orientation of the key-point. Two points are considered a match if the error of the overlay of their corresponding regions overlay is less than 40%. This metric is widely used for 2D descriptors and is available in the OpenCV library.

5.4.3. Hough Transform Implementation Impact

One of the first decisions that had to be made regarding the light-field detector was the selection of a specific Hough transform implementation. Two options were considered: the OpenCV implementation

[58], and the scikit-image toolbox [60]. The OpenCV implementation provides options such as built–in filtering by a minimum number of points and choosing ρ and Θ sensitivity parameters while the scikit-image implementation does not. However, after experimenting with different ρ and Θ sensitivity values it was concluded that the OpenCV implementation is not ideal for this type of application since its results were not those expected with the provided input parameters, i.e., when the ρ and Θ parameters are changed (i.e. increasing its values), more repeated straight lines were detected, while the opposite behaviour was expected. Figure 57 shows this phenomenon where setting the ρ and Θ sensitivity to 4 pixels and 4 degrees, respectively, rather than ρ=0.5 pixels and Θ=0.5 degrees, led to more lines detected around the same region. This means that despite an increase of the ρ and Θ parameters (which lead to a coarse quantization of the parameters and thus a 2D grid with less bins) more straight lines were detected. These results led to the choice of the scikit-image Hough transform implementation instead, which, as displayed in Figure 58, shows the expected behaviour, since the number of detected lines decreases as ρ and Θ are increased. Note that the results in both figures were obtained before any line filtering was performed (check the EPI Line Detection module in Section 5.1).

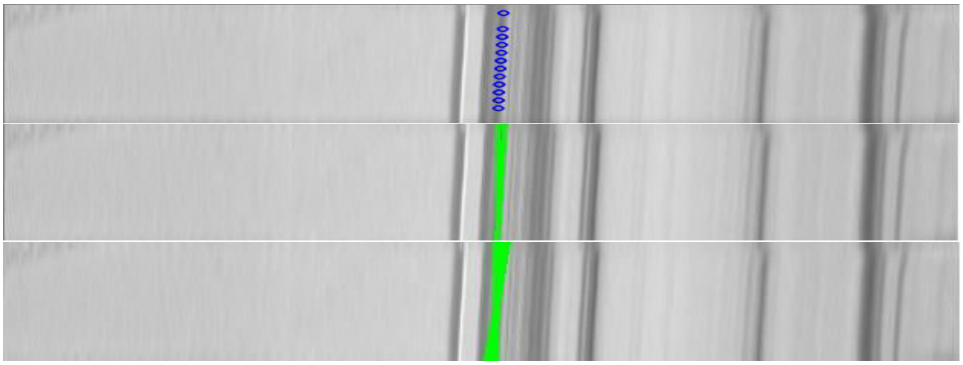

Figure 58: Detected Hough lines using OpenCV Hough Transform. Top: Key-points projected to EPI. Middle: Detected lines with sensibility parameters ρ=0.5 pixels and Θ=0.5 degrees. Bottom: Detected lines with sensibility parameters ρ=4 pixels and Θ=4 degrees.

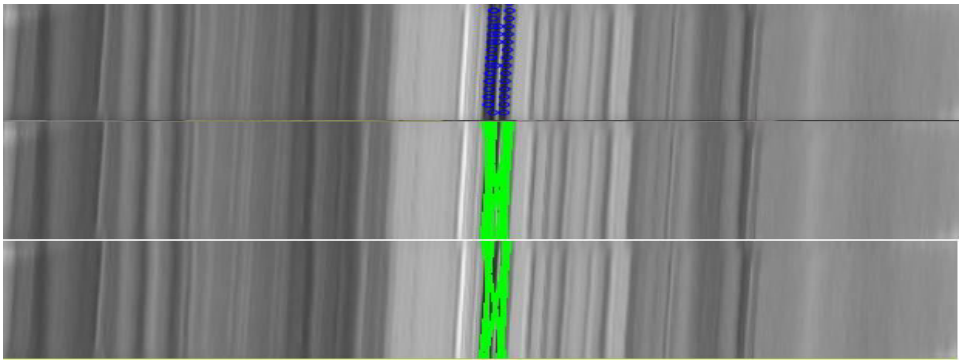

Figure 59: Detected Hough lines using scikit-image Hough Transform. Top: Key-points projected to EPI. Middle: Detected lines with sensibility parameters ρ=1 pixels and Θ=1 degrees. Bottom: Detected lines with sensibility parameters ρ=8 pixels and Θ=8 degrees.

5.4.4. Minimum Number of Key-Location Points Impact

Another important implementation decision to make was the selection of the minimum number of 2D key-points that a single key-location should have to be retained (and thus not filtered). This filtering threshold is here designated as M_p; note that $1 \leq M_p \leq 15$, since there are 15 images in each vertical or horizontal array.

The evaluation framework presented in Section 5.2.2 was used to assess the influence of this parameter on the overall performance of the light-field key-location detector. In this case, the detector was applied with different M_p values, and the results compared side-by-side with the SIFT 2D key-point detector applied to the central sub-aperture image of each light-field. SIFT was selected as benchmark since it is part of our light-field key-location detector, and it is arguably one of the more stable key-point detectors currently available. The M_p values that are too low (<4) or too high (>12) are not considered, since these were proven in earlier tests to gravely affect the performance of the detector.

Table 10 shows the repeatability results for different M_p, using R_{pos}, and retaining the same number of key-points/key-locations on both sides, i.e. the number of key-locations output by the detector is used as a parameter for the number of retained key-points in the SIFT detector (this parameter will also be optimized in a later section). The two columns for each M_p indicate the average repeatability value (for all transformation pairs) for the light-field key-location detector (on the left, LF KL Detector), and the SIFT 2D key-point detector (on the right, SIFT KP Detector); each cell contains the average repeatability over two pairs of images, for each transformation. Values in **bold** indicate which solution had the best repeatability performance for each individual case. As shown, the proposed light-field key-location detector solution reaches its best performance for values between $6 \leq M_p \leq 8$ showing a slight edge over SIFT for those values. This means that our detection solution works best when it considers key-points detected in about half the images of the sub-aperture array. Therefore, the remaining performance assessment tests were performed with $M_p = 7$.

Transformation	Mp=4		Mp=6	
	LF KL Detector	SIFT KP Detector	LF KL Detector	SIFT KP Detector
Scaling	0.598	**0.601**	0.609	**0.634**
Rotation	**0.772**	0.758	**0.779**	0.744
Viewpoint Change	**0.579**	**0.579**	**0.579**	0.577
Blur	**0.653**	0.649	**0.656**	0.645
Transformation	Mp=7		Mp=8	
	LF KL Detector	SIFT KP Detector	LF KL Detector	SIFT KP Detector
Scaling	**0.603**	0.574	**0.574**	0.562
Rotation	**0.784**	0.74	**0.775**	0.745
Viewpoint Change	0.572	**0.579**	0.563	**0.573**
Blur	**0.656**	0.638	**0.652**	0.641
Transformation	Mp=10		Mp=12	
	LF KL Detector	SIFT KP Detector	LF KL Detector	SIFT KP Detector
Scaling	**0.521**	0.515	0.46	**0.508**
Rotation	0.7	**0.724**	0.628	**0.782**
Viewpoint Change	0.539	**0.545**	0.477	**0.544**
Blur	**0.629**	0.627	0.606	**0.676**

Table 10: Repeatability results for different M_p *values, for Repeatability criterion 1 (*R_{pos}*).*

5.4.5. Key-Location Selection Strategy Impact

The number of key-locations obtained varies from light-field to light-field. However, it may be useful to obtain a fixed number of key-locations for each light-field to apply feature extraction (later) and obtain a description with the same amount of information, e.g. with the same number of descriptors independently of the image. Naturally, the key-locations selected should be the best according to some criterion. This step is essential to perform the final evaluation of the light-field key-location detector and requires the classification of each key-location with some level of importance. Usually, the 2D feature detectors compute a response value that represents the relevance of each key-point and only the descriptors associated to the key-points with the highest response are extracted. The sorting of the key-locations can be performed according to some relevance metric, and by doing so, enabling the user to choose how many key-locations are kept, and providing a fair comparison with the SIFT 2D detector. The three metrics evaluated to sort the key-locations are:

- **Sorting by response**: This first option is the same as usually performed when SIFT detectors are used. The main difference is that the response associated to our key-locations is the average of the various 2D key-point responses corresponding to that specific key-location. The response of a key-point is obtained through the computation of the determinant of its second order Hessian Matrix, which quantizes the local changes in illumination around the key-point location.

- **Sorting by score and response**: The second option takes into consideration the score of each key-location i.e. the number of its key-points and multiplies this value by its response, the response parameter is described above.

- **Sorting by depth**: This third option sorts the key-locations by their computed depth value; the closer to the camera, the higher it is ranked.

Table 11 shows the repeatability using R_{pos} results for the three proposed sorting methods, for a total of 100 key-locations. Each cell value is the average repeatability over two transformation pairs. The values in **bold** indicate the best performing sorting solution for each case. By analysing the results, it is clear that the best approach is to sort the key-locations by its response value since this criterion outperforms the other alternatives, even managing to outperform SIFT in many cases.

Transformations	Sorting by score and response	Sorting by response
Scaling	0.34	**0.475**
Rotation	0.585	**0.81**
Viewpoint Change	0.35	0.405
Blur	0.56	**0.575**
Transformations	**Sorting by depth**	**SIFT**
Scaling	0.24	0.45
Rotation	0.26	0.685
Viewpoint Change	0.185	**0.435**
Blur	0.215	0.515

Table 11: Repeatability results for different sorting methods.

5.5. Final Detection Performance Evaluation

This final performance evaluation was performed to compare the proposed detector solution with the SIFT 2D detector when the number of retained key-locations is changed. This will allow a full evaluation of the proposed light-field key-point detector for different levels of description strength. For these tests, two more image pairs were added for each transformation, and the second repeatability metric R_{all}, widely used for 2D descriptor was used in addition to R_{pos}. The full range of retained key-locations was varied from 100 to 450 unique key-locations per image. Table 12 shows the results obtained with R_{pos}, when the number of retained key-locations varies from 150 to 300 (the full table will be available in the annexes); this variation corresponds to the range for which the light-field key-point detector has the best detection results. Values in each cell indicate the average repeatability value over 4 different transformation pairs. Cells on the left side of the table indicate results for the light-field key-location detector, while values in the middle indicate results for SIFT. The right-most column indicates the gains obtained for each case. For this metric, the light-field key-location detector almost always out-performs SIFT, although the performance gains are generally not very large. Using the R_{pos} metric, rotation and scaling are the image transformation where our solution seems to present larger performance gains, up to 4 percentage points; for blurring and viewpoint-change, the performance gains are around 1 to 2 percentage points. For the 200 points case, the light-field key-location detector seems to have lower performance than SIFT, specifically for viewpoint-change, notably by around 1 percentage point.

Transformations	**300 Locations/Points**		**Performance Gains**
	LF KL Detector	**SIFT KP Detector**	
Scaling	**0.537**	0.518	0.019
Rotation	**0.728**	0.698	0.03
Viewpoint Change	**0.545**	0.534	0.011
Blur	**0.623**	0.606	0.017
Transformations	**250 Locations/Points**		**Performance Gains**
	LF KL Detector	**SIFT KP Detector**	
Scaling	**0.536**	0.517	0.019
Rotation	**0.729**	0.691	0.038
Viewpoint Change	**0.531**	0.521	0.01
Blur	**0.622**	0.615	0.007
Transformations	**200 Locations/Points**		**Performance Gains**
	LF KL Detector	**SIFT KP Detector**	
Scaling	**0.529**	0.491	0.038
Rotation	**0.713**	0.681	0.032
Viewpoint Change	0.506	**0.521**	-0.015
Blur	**0.605**	0.578	0.027
Transformations	**150 Locations/Points**		**Performance Gains**
	LF KL Detector	**SIFT KP Detector**	
Scaling	**0.505**	0.463	0.042
Rotation	**0.702**	0.663	0.039
Viewpoint Change	**0.502**	0.495	0.007
Blur	**0.588**	0.571	0.017

Table 12: Repeatability results for Rpos metric using different numbers of retained key-locations/key-points.

Table 13 contains the same results using the R_{all} metric, it becomes clear by analysing it that the behaviours of both the light-field key-point detector and the SIFT 2D detector are not as stable as when using R_{pos}, since the behaviours vary not only according to the number of retained unique key-locations/key-points, but also according to which image transformation the repeatability is assessed. It is also clear that the light-field key-location detector generally outperforms SIFT, although not as consistently as when using R_{pos}. In this case, it is pertinent to perform a more detailed analysis of results. Analysing the table according to each distinct transformation, the following observations are possible:

- **Blur**: For this transformation, as expected, the light-field detector doesn't show a major advantage since the inclusion of additional sub-aperture images capturing the light in different directions is not beneficial, since the extra information provided by the disparity between different sub-aperture images is not relevant in the case of blur transformations. For the case of blurring, the 2D SIFT key-point detector seems to have a slight (but not very relevant) advantage.

- **Viewpoint change**: For the case of viewpoint-changes, the light-field key-location detector has a considerable advantage over SIFT, which was expected. The advantage is the largest for 150-200 retained key-locations, where it reaches 10 percentage points improvement.

- **Rotation**: Rotation is another transformation where the light-field key-point detector should also bring performance gains. For this range of retained unique-key-locations/key-points, the light-field key-location detector was never outperformed by SIFT, and always being able to maintain a score of around 6 percentage points higher than SIFT.

- **Scaling**: Although results for scaling were not as positive and stable as for rotations, they still show that the light-field key-location detector, although being occasionally outperformed by SIFT (as it is the case for 300 retained key-locations), it manages to have an edge over it for lower numbers of retained key-locations/key-points, being able to outperform SIFT by 4 to 5 percentage points.

Transformations	300 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	0.562	**0.579**	-0.017
Rotation	**0.682**	0.622	0.06
Viewpoint Change	**0.58**	0.492	0.088
Blur	0.444	**0.47**	-0.026
Transformations	**250 Locations/Points**		**Performance Gains**
	LF KL Detector	**SIFT KP Detector**	
Scaling	**0.601**	0.561	0.04
Rotation	**0.69**	0.614	0.076
Viewpoint Change	**0.562**	0.535	0.027
Blur	0.457	**0.472**	-0.015
Transformations	**200 Locations/Points**		**Performance Gains**
	LF KL Detector	**SIFT KP Detector**	
Scaling	**0.6**	0.56	0.04
Rotation	**0.679**	0.613	0.066
Viewpoint Change	**0.571**	0.473	0.098
Blur	**0.459**	0.454	0.005
Transformations	**150 Locations/Points**		**Performance Gains**
	LF KL Detector	**SIFT KP Detector**	
Scaling	**0.598**	0.543	0.055
Rotation	**0.669**	0.604	0.065
Viewpoint Change	**0.559**	0.45	0.109
Blur	0.439	**0.456**	-0.017

Table 13: Repeatability results for *Rall* metric using different numbers of retained key-locations/key-points.

From these results, it is possible to show that the ideal number of retained unique key-locations to be used is between 200 and 250, since for this range higher repeatability values are obtained, while also maintaining a considerable advantage over SIFT. As a final note, it is important to state that these repeatability values were the result of tests on uncontrolled image transformations aside from blur. This inevitably affects the performance of any key-point detector in comparison to other methods used for evaluation in the literature which apply an artificial transformation to obtain one image from another [64]. .

Chapter 6

6. Light-Field Key-Point Description

This chapter describes the proposed solution for a light-field local descriptor which was developed to exploit the rich information of lenslet light-fields but can ideally be extended to other light-field formats. This light-field local descriptor follows the light-field key-point detector presented in the previous chapter, and therefore uses the output of that same detector i.e. the light-field key-locations as a starting point. This descriptor is scalable in the sense that can be used to characterize the light field image but also perform matching with a 2D descriptors with a base layer that is SIFT compliant. The enhancement layer of the light-field descriptor is used to characterize another dimension which is across the several sub-aperture images that are available in the light field format, both horizontally as well as vertically. This chapter is structured as follows, first the framework of the base layer of the light field local descriptor is presented. Next, some results for the base layer are presented and compared to other local 2D descriptors. After an extension of this base layer is described, i.e. an enhancement layer. This extension takes into account extra information provided by the light-field detector such as the slope of the straight line defined by each key-location to improve descriptor distinctiveness and robustness. Finally, some results for this extension are presented.

6.1. Light-Field Key-Point Descriptor Base Layer Framework

This section presents a walkthrough of the base layer of our light-field key-point descriptor, Figure 59 presents the framework of the descriptor.

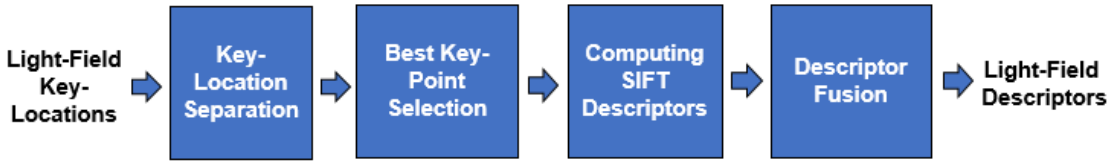

Figure 60: Light-field key-point descriptor base layer framework.

This framework can be divided into the following distinct steps:

1. **Key-Location Separation**: As defined in the previous chapter, key-locations are data structures containing a set of 2D key-points, as well as the parameters relative to the line defined by that same set of key-points. The set of key-points contained in each key-location is composed of key-points which were detected at the same spatial coordinates in different sub-aperture images of either the horizontal or vertical array of images. However, since the foundation of our light-field key-

location is the SIFT detector, it is possible to have two key-points at the exact same coordinates but with different orientations; this is a critical detail, since this means that the same set of spatial coordinates can contain information regarding several visual features instead of just one, hence it is essential to adapt our descriptor to be able to correctly process points at the same spatial coordinates but with distinct orientations. This phenomenon happens in the following way: During the key-point orientation assignment phase of the SIFT detector, a 36 bin orientation histogram is computed to find the dominant orientation in the patch (detected region), i.e. the angle which displays the strongest magnitude of the histogram. If during this orientation phase, the magnitude of another bin is higher than 80% of the magnitude of the bin that represents the dominant orientation, a second key-point is created with the corresponding orientation. Figure 60 illustrates this case.

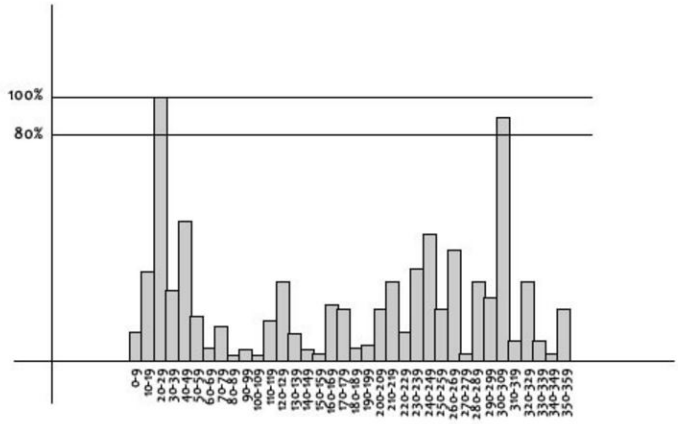

Figure 61: SIFT local key-point separation [15].

Therefore, it is necessary to separate possible key-points which might be at the same coordinates, but with distinct orientations, since orientation is a fundamental component of any rotation-invariant descriptor. Any key-location which contains key-points with distinct orientation values is separated into several key-locations, each one containing only one orientation value. To further illustrate this situation, Figure 61 shows a rendered central view of one of the light-fields in the Lisbon landmark dataset with its respective detected key-points. In this figure, the radius of the circumference around each key-point is proportional to its size, while the direction of the straight line drawn from the center to the edge of the circumference indicates the orientation of the key-point. As shown, some of these points have multiple orientations.

Figure 62: Detected key-points over Lisbon landmark light-field dataset image.

2. Best Key-Point Selection: The second step is to select the key-point inside each key-location which is closest to the central image of each array. The sub-aperture images of the Lytro lenslet Light-fields often display stronger optical aberrations the further the sub-aperture image is from the centre of the array. Therefore, the key-points detected closest to the central images of each array (horizontal or vertical) are more likely to be the more reliable, i.e. have the most reliable attributes, therefore the size, orientation, response, and octave of the key-point inside each key-location which is closest to the central sub-aperture image are stored, these attributes are representative of the entire key-location and every descriptor computation that this framework applies in the following steps is performed using these attributes, but centered at varying spatial coordinates.

3. Computing SIFT descriptors: The third step is to apply the SIFT feature extraction method (as described in chapter 2) to each selected key-location at nearly every sub-aperture image of the array where that key-location was detected, the first and last images of both the horizontal and vertical arrays are discarded, since they are somewhat distorted, and therefore would negatively affect the feature extraction process. If the key-location was detected in the horizontal EPI array, a SIFT descriptor is computed for the images in the horizontal sub-aperture image array using the attributes of the key-point selected in step 2, i.e. the key-point within the key-location that is closest to the central sub-aperture image, The same applies to the vertical array. Note that the values of these attributes heavily affect the SIFT descriptor since SIFT was designed to be scale and rotation invariant, and this invariance relies on these attributes, rotation invariance is achieved by rotating the image patch for which the SIFT descriptor is computed according to the orientation value of its corresponding key-point as well as by adjusting the angles of the gradients inside each patch according to that same orientation, while scale invariance is achieved by resizing that same patch according to the size and octave of its corresponding key-point. To take into account the disparity between sub-aperture images, the displacement that a visual feature presents from sub-aperture image to sub-aperture image is computed through the angle of the straight line contained in the key-location corresponding to that visual feature, and its coordinates are adjusted accordingly. This displacement is used to shift the coordinates of the center of the SIFT descriptor along the array of

sub-aperture images. To determine which spatial coordinates vary from sub-aperture image to sub-aperture image, one only needs to know to which sub-aperture image array the key-location belongs to, i.e. if the key-location belongs to the horizontal array, then the varying spatial coordinate will be *x*, if it belongs to the vertical array, then the varying coordinate will be *y*. After determining which spatial coordinate is varying, the amount of variation per sub-aperture image can be computed through the following geometric formula:

$$dis = \frac{1}{\tan(90-\theta)} \qquad (6.\ 1)$$

Where dis is the variation, Θ is the angle of the key-location, the denominator is of value 1 since we are computing the displacement per sub-aperture image, if one would compute the total variation the value 1 would be replaced by the value 15, since there are 15 sub-aperture images per array. This assures that the position of each key-point relative to each sub-aperture image is computed as precisely as possible.

4. Descriptor Fusion: The fourth and final step is to average the descriptors computed in the previous step for each key-location across the sub-aperture image array, in order to obtain a single, 128 float sized descriptor, which contains information on every sub-aperture image in the array.

6.2. Light-Field Key-Point Descriptor Extension Framework

This section presents an extension of the previous descriptor to further take advantage of the lenslet light-field format. The output of this framework is a second descriptor which is added to the base layer descriptor. This brings several advantages, such as the matching between 2D descriptors and LF descriptors as well the matching between LF descriptors. This second descriptor adds information by exploiting local illumination changes across the EPI arrays**.** Figure 62 shows the framework of this extension.

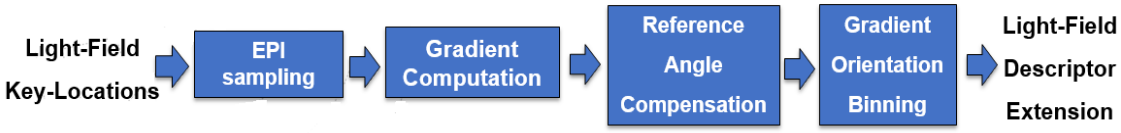

Figure 63: Light-field key-point descriptor extension framework.

The walkthrough of this framework is the following:

1. EPI sampling: This framework receives as input a light-field key location. This is a structure containing the parameters defining a straight line obtained in the detection phase (ρ, Θ, depth and score), as well as the parameters of each key-point contained in the detected line (coordinates, size, orientation…). The first step is to sample the EPI's around the coordinates of the light-field key-location, to do this the EPI containing the key-location added to the 8 EPI's surrounding the coordinates of the key-location (in both directions) are sampled. Within each sampled EPI, the region surrounding the light-field key-location is also sampled, leading to a total of 17 sampled

regions of interest, these sets of regions of interest contain visual information on several different sub-aperture images, and therefore they can be called 3D patches. Figure 63 shows an illustration of this sampling process.

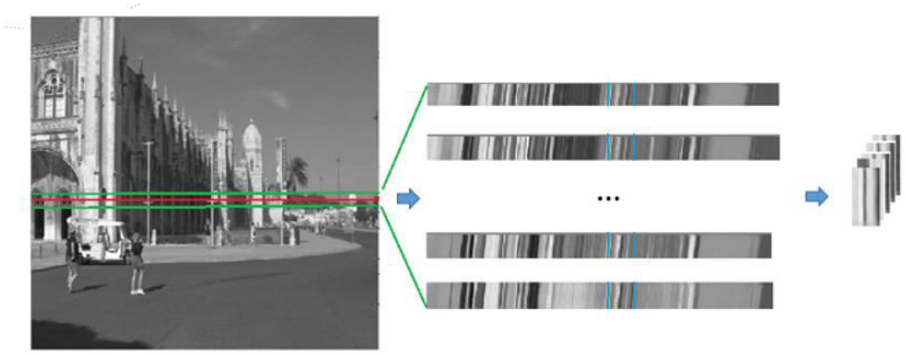

Figure 64: Extracting 3D patch from adjacent EPIs.

2. Gradient Computation: Next, for each 3D patch, a gradient computation is performed. This is done using a Sobel operator on each of the sampled ROI which compose the 3D patch, the Sobel operator extracts gradient values point to point using equation 6.2, where *G* indicates a gradient value and *L* indicates a luminance value at a given set of coordinates. This is done to characterize the intensity changes that occur within each region of interest. The magnitude and orientation of each gradient within the 3D patch is computed for later processing.

$$G(x, y) = L(x, y + 1) - L(x, y - 1) \qquad (6.2)$$

3. Reference Angle Compensation: Having computed gradient magnitudes and orientations for the 3D patch, an angle compensation is performed. The angle used for compensation is that of the straight line defining the corresponding key-location, this angle is subtracted to the gradient orientations within each 3D patch, this is done to compensate for the different depth values of each key-location (since the slope of the straight line of each key-location is proportional to its depth), and ideally provide depth invariance to this light-field descriptor extension.

4. Gradient Orientation Binning: Having performed angle compensation, the final step is to perform binning on the gradient orientations within each 3D patch, in order to describe it in a compact way. For this binning process, an uneven set of bins was selected. This choice was made because the EPI's extracted for light-fields within our dataset display a somewhat narrow range of possible angles, and therefore the angles of the detected key-locations also fall within this narrow range. Therefore, for angles between [-45, 45] degrees, a narrow 2 degree window was selected, and for angles outside this region, a wide, 10 degree window was selected. This leads to an orientation histogram of 45 narrow bins, and 27 wide bins. Hence a histogram of a total of 72 bins. Each time a gradient orientation value falls within a given bin, that same bin is incremented with the gradient's corresponding magnitude value. Another important implementation detail regarding this step is how to group the regions of interest within each 3D patch. For our implementation, it was decided that four separate 72 binned histograms would be used. Three histograms covering groups

of 4 regions of interest, and one histogram covering a group of 5 regions of interest (which also includes the central EPI, hence the extra region). This leads to a 72*4= 288 sized light-field descriptor extension. This extension is admittedly large, and should eventually undergo a fine tuning process, in which the histogram bin ranges should be adjusted, in order to make it more efficient. A final note to add to this framework is that considering the large size of this extension, relative to its base layer, a multiplication factor was applied to the extension layer of the descriptor. This multiplication factor assures that more emphasis is given on the base layer of the descriptor, relative to the extension. Upon testing a wide array of possible values, it was discovered that a multiplication factor 0.4 produced the best performance results.

6.3. Performance Evaluation

To validate the proposed solution a random set of 25 queries was selected from both the restricted landmark dataset (one query per landmark), and the full landmark dataset, and matching was performed. The matching method and evaluation metrics used are the ones described section 4.3.1, using CVM (central view matching) for the 2D descriptors. The proposed solution was compared to 2D descriptors when applied to the central image of the light-field array. Although other benchmarks could be used, most of the other possibilities require the concatenation of several different descriptors and/or a modified matching criteria. To guarantee a fair evaluation it is necessary to evaluate descriptors with the same length, i.e. the compactness of the descriptor should be the same for all solutions being evaluated. Table 14 shows the results obtained.

MAP		
Descriptors	**Full Dataset**	**Controlled Dataset**
SIFT	35.2	57.3
SURF	34	53
ORB	27	46
BRIEF	15	39.3
Light-Field Descriptor - Base Layer	36.1	58.4
Light-Field Descriptor - Base Layer+ Extension	**36.5**	**58.9**

Table 14: Mean average precision results for the LF descriptor, compared to 2D descriptors.

By comparing the different metrics for each descriptor, it is clear that the proposed LF descriptor has better performance when compared to SIFT and other 2D descriptors. Although these gains are not very large (slightly above 1 percentage point), they provide proof that using information across different sub-aperture images of one light-field can increase matching performance. Naturally, if more sub-aperture image arrays were used, i.e. not just the central vertical and horizontal arrays or if light-fields with a wider disparity between sub-aperture images are used the matching performance would considerably increase. Regarding the light-field descriptor extension, slight gains were also obtained relative to the initial base-layer, this proves that adding visual information description obtained directly from the sampled EPI's can help to improve descriptor performance even further. Once again we predict that the method proposed for this extension would be bring larger gains if a light-field dataset containing larger disparity between sensors was used. We also predict that additional gains would be obtained if sensors with larger resolution, and better quality were used.

Chapter 7

7. Conclusions and Future Work

This Thesis studies the problem of feature detection and extraction for the light-field visual representation format, and proposes a novel light-field image retrieval dataset as well as new image matching methods to exploit existent 2D descriptors with this format. After, a local feature detector and extractor tailored specifically for light-fields is proposed, namely by exploiting horizontal and vertical arrays (cubes) of Epipolar Plane Images.

To fully understand the details and intricacies of this field, a detailed state-of-the-art review on both the topics of feature detection/extraction and plenoptic imaging was performed. The contents of this review serve both as a theoretical introduction on the subject and as the starting point for the following development phase.

The first part of the development phase was the design and acquisition of the Lisbon light-field landmark dataset, a dataset specifically tailored for image retrieval using light-fields. The dataset contains light-fields portraying 25 distinct landmarks from the city of Lisbon where each landmark was captured under different viewing conditions, containing several image transformations such as viewpoint changes, in-plane rotations, and detail shots. In order to study the effect of more difficult image transformations on retrieval performance this dataset was divided into a controlled, and full version. While the full version contains every light-field in the dataset, the controlled version excludes the image transformations considered to be the most “difficult”. Also, a few novel matching methods were proposed that take advantage of the extra visual information contained in lenslet light-fields and standard 2D descriptors are applied. By combining information from several different sub-aperture images contained in the sub-aperture image array encoded in the lenslet light-field, it was able to considerably improve retrieval performance when compared to standard 2D images.

The second part of the development phase of this Thesis introduces a key-point detector adapted to light-fields. This detector uses cubes of Epipolar Plane Images (EPI) contained in the central horizontal, and central vertical arrays of sub-aperture images to define so-called “key-locations”, which are data structures containing several key-points corresponding to the same visual features, but detected at different sub-aperture images. In addition to these key-points, information regarding the straight line formed by these key-points in the EPI is also encoded into each key-location. By properly processing these key-locations is possible to determine which visual features contained in a light-field are the most important, and sort them accordingly, therefore improving the feature detection process. This light-field key-point detector was assessed and compared to the SIFT key-point detector through

the use of a evaluation framework and robust repeatability metrics. The results show that the proposed light-field detector surpasses SIFT for several image transformations such as viewpoint-change and rotation.

The third and final part presents a local descriptor adapted to light-fields. This descriptor is paired together with the proposed light-field detector, and can extract more distinctive visual features by using the information encoded in each key-location. This is performed by changing the spatial coordinates where a SIFT descriptor is computed along each sub-aperture image in the central horizontal, or central vertical sub-aperture image arrays, depending on which of these arrays the corresponding key-location belongs to. The change of spatial coordinates along each image of the array is proportional to the slope of each key-location obtained in the light-field feature detection phase. The resulting light-field descriptor is then an average of every 2D descriptor computed for the same visual feature along an array of sub-aperture images. An additional extension layer was added to this light-field descriptor, which by more directly describing visual information present in EPI's, lead to further performance gains being obtained. This solution was evaluated according to well-known precision metrics for image matching such as Mean Average Precision, and Top-Level Precision, and compared to the some of the most high-performance 2D descriptors such as SIFT and SURF. The results obtained show that the proposed descriptor is indeed able to improve the 2D feature detection methods. This method can also be applied to other light-fields formats (e.g. obtained from 2D array of cameras) apart from the lenslet light-field and it is predicted that better performance gains can be achieved when larger horizontal and vertical disparity is available from the set of multiple perspectives that capture different light directions.

Regarding future work, it would be interesting to test and adapt the proposed light-field feature detector and extractor to other light-field formats, since the performance for image retrieval would likely improve, especially when larger disparity is available among the different perspectives (or sub-aperture images).

A possible improvement of the proposed light-field feature detector and extractor is to explore the lenslet light-field in other ways. Our solutions only take into account a fraction of the visual information available inside each lenslet light-field: the central horizontal and central vertical arrays of sub-aperture images. Therefore, it is expected to obtain some gains in performance by also exploiting other sub-aperture images other than these horizontal and vertical 3D arrays.

Finally, another possible research topic is the development of a binary feature descriptor for light-field images. While real-valued feature descriptors tend to produce the best performance results when it comes to image retrieval, binary descriptors have the advantage of being more compact, and generally faster to compute, and faster to match. To our knowledge, no binary feature descriptor for light-fields has yet been developed. Hence this would also be an interesting line of work to pursue in the field of visual feature representations.

Annex A

Transformations	450 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	**0.555**	0.538	0.017
Rotation	**0.752**	0.726	0.026
Viewpoint Change	**0.588**	0.547	0.041
Blur	**0.647**	0.542	0.105
Transformations	400 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	**0.552**	0.533	0.019
Rotation	**0.728**	0.721	0.007
Viewpoint Change	0.564	**0.568**	-0.004
Blur	**0.63**	0.614	0.016
Transformations	350 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	**0.547**	0.522	0.025
Rotation	**0.736**	0.711	0.025
Viewpoint Change	**0.553**	0.544	0.009
Blur	**0.622**	0.51	0.112
Transformations	300 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	**0.537**	0.518	0.019
Rotation	**0.728**	0.698	0.03
Viewpoint Change	**0.545**	0.534	0.011
Blur	**0.623**	0.606	0.017
Transformations	250 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	**0.536**	0.517	0.019
Rotation	**0.729**	0.691	0.038
Viewpoint Change	**0.531**	0.521	0.01
Blur	**0.622**	0.615	0.007
Transformations	200 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	**0.529**	0.491	0.038
Rotation	**0.713**	0.681	0.032
Viewpoint Change	0.506	**0.521**	-0.015
Blur	**0.605**	0.578	0.027
Transformations	150 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	**0.505**	0.463	0.042
Rotation	**0.702**	0.663	0.039
Viewpoint Change	**0.502**	0.495	0.007
Blur	**0.588**	0.571	0.017
Transformations	100 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	**0.49**	0.455	0.035
Rotation	**0.685**	0.598	0.087
Viewpoint Change	**0.633**	0.59	0.043
Blur	**0.55**	0.51	0.04

Table 16: Full repeatability results for *Rpos*.

Annex B

Transformations	450 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	0.557	**0.619**	-0.062
Rotation	**0.702**	0.649	0.053
Viewpoint Change	**0.608**	0.542	0.066
Blur	**0.449**	0.441	0.008
Transformations	400 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	0.558	**0.598**	-0.04
Rotation	**0.698**	0.648	0.05
Viewpoint Change	**0.596**	0.56	0.036
Blur	0.439	**0.454**	-0.015
Transformations	350 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	0.552	**0.618**	-0.066
Rotation	**0.692**	0.638	0.054
Viewpoint Change	**0.592**	0.552	0.04
Blur	**0.45**	0.446	0.004
Transformations	300 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	0.562	**0.579**	-0.017
Rotation	**0.682**	0.622	0.06
Viewpoint Change	**0.58**	0.492	0.088
Blur	0.444	**0.47**	-0.026
Transformations	250 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	**0.601**	0.561	0.04
Rotation	**0.69**	0.614	0.076
Viewpoint Change	**0.562**	0.535	0.027
Blur	0.457	**0.472**	-0.015
Transformations	200 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	**0.6**	0.56	0.04
Rotation	**0.679**	0.613	0.066
Viewpoint Change	**0.571**	0.473	0.098
Blur	**0.459**	0.454	0.005
Transformations	150 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	**0.598**	0.543	0.055
Rotation	**0.669**	0.604	0.065
Viewpoint Change	**0.559**	0.45	0.109
Blur	0.439	**0.456**	-0.017
Transformations	100 Locations/Points		Performance Gains
	LF KL Detector	SIFT KP Detector	
Scaling	0.519	**0.556**	-0.037
Rotation	**0.635**	0.564	0.071
Viewpoint Change	**0.454**	0.425	0.029
Blur	**0.435**	0.43	0.005

Table 17: Full repeatability results for *Rall*.

Bibliography

[1] D. G.Lowe, “Distinctive Image Features from Scale-Invariant Keypoints,” *International Journal of Computer Vision,* vol. 60, pp. 91-110, October 2004.

[2] H. Bay, T. Tuytelaars and L. Van Gool, “SURF: Speeded Up Robust Features,” in *European Conference on Computer Vision*, Graz, Austria, May, 2006.

[3] S. Leutenegger, M. Chli and R. Y. Siegwart, “BRISK: Binary Robust Invariant Scalable Keypoints,” in *International Conference on Computer Vision*, Barcelona, Spain, November, 2011.

[4] “KZMZ News,” [Online]. Available: http://totalitec.blogspot.pt/2015/12/facebook-messenger-introduces-face.html. [Accessed May 2016].

[5] “Google Images,” [Online]. Available: https://images.google.com/. [Accessed May 2016].

[6] “Pixolution Visual Image Search,” [Online]. Available: http://demo.pixolution.de/. [Accessed May 2016].

[7] E. H. Adelson and J. R. Bergen, “The Plenoptic Function and the Elements of Early Vision,” in *Computational Models of Visual Processing*, MIT Press, 1991, pp. 3-20.

[8] “Lytro Official Website,” [Online]. Available: https://www.lytro.com/. [Accessed 2016 May].

[9] “Raytrix Official Website,” [Online]. Available: https://www.raytrix.de/. [Accessed May 2016].

[10] A. Ghasemi and M. Vetterli, “Scale-Invariant Representation of Light Field Images for Object Recognition and Tracking,” in *Computational Imaging XII*, vol. 9020, SPIE, March, 2014.

[11] I. Tosic and K. Berkner, “3D Keypoint Detection by Light Field Scale-Depth Space Analysis,” in *International Conference on Image Processing*, Paris, France, October, 2014.

[12] R. Fergus, “Corners, Blob & Descriptors,” [Online]. Available: http://cs.nyu.edu/~fergus/teaching/vision_2012/3_Corners_Blobs_Descriptors.pdf. [Accessed Aprl 2016].

[13] D. Kent, “Georgia College of Tech Computing,” [Online]. Available: http://www.cc.gatech.edu. [Accessed April 2016].

[14] F. Pereira, “Feature-Based Representation Requirements,” in *European Signal Processing*

Conference, Marrakech, Morocco, September, 2013.

[15] “Introduction to SIFT,” [Online]. Available: http://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/. [Accessed March 2016].

[16] M. Agrawal, K. Konolige and M. R. Blas, “CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching,” in *European Conference on Computer Vision*, Marseille, France, October, 2008.

[17] E. Rosten, “FAST Corner Detection -- Edward Rosten,” [Online]. Available: http://www.edwardrosten.com/work/fast.html. [Accessed April 2016].

[18] E. Rosten, R. Porter and T. Drummond, “Faster and Better: A Machine Learning Approach to Corner Detection,” in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008, pp. 105-119.

[19] E. Mair, G. D. Hager, D. Burschka, M. Suppa and G. Hirzinger, “Adaptive and Generic Corner Detection Based on the Accelerated Segmented Test,” in *European Conference on Computer Vision*, Crete, Greece, September, 2010.

[20] N. Dalal and B. Triggs, “Histogram of Oriented Gradients for Human Detection,” in *Conference on Computer Vision and Pattern Recognition*, San Diego, CA, USA, June, 2005.

[21] J. T. Pedersen, “Study group SURF: Feature Detection & Description,” [Online]. Available: http://cs.au.dk/~jtp/SURF/report.pdf.

[22] “Pattern recognition systems- Lab 5 Histogram of Oriented Gradients,” [Online]. Available: http://users.utcluj.ro/~raluca/prs/prs_lab_05e.pdf. [Accessed April 2016].

[23] M. Calonder, V. Lepetit, C. Strecha and P. Fua, “BRIEF: Binary Robust Independent Elementary Features,” in *European Conference on Computer Vision*, Crete, Greece, September, 2010.

[24] A. Alahi, R. Ortiz and P. Vandergheynst, “FREAK: Fast Retina Keypoint,” in *Conference on Computer Vision and Pattern Recognition*, Providence, RI, USA, June, 2012.

[25] “Gil's CV Blog,” [Online]. Available: https://gilscvblog.com. [Accessed April 2016].

[26] E. Rublee, V. Rabaud, K. Konolige and G. Bradksi, “Orb: An Efficient Alternative to SIFT or SURF,” in *International Conference on Computer Vision*, Barcelona, Spain, Novemeber, 2011.

[27] A. Canclini, A. Cesana, A. Redondi, J. Ascenso, M. Tagliasacchi and R. Cilla, “Evaluation of Low-Complexity Visual Feature Detectors and Descriptors,” in *International Conference on Digital Signal Processing*, Santorini, Greece, July, 2013.

[28] K. Kitani, “Bag-of-Visual-Words,” [Online]. Available: http://www.cs.cmu.edu/~16385/lectures/Lecture12.pdf. [Accessed April 2016].

[29] “A Bayesian Hierarchical Model for Learning Natural Scene Categories",” in *Conference on Computer Vision and Pattern Recongnition*, Pasadena, CA, USA, June, 2005.

[30] “Wikipedia on Bag-of-Words Model,” [Online]. Available: https://en.wikipedia.org/wiki/Bag-of-words_model. [Accessed April 2016].

[31] “Wikipedia on k-means clustering,” [Online]. Available: https://en.wikipedia.org/wiki/K-means_clustering. [Accessed April 2016].

[32] J. Kundrac, “Bag of visual words in OpenCV,” [Online]. Available: http://vgg.fiit.stuba.sk/2015-02/bag-of-visual-words-in-opencv/. [Accessed April 2016].

[33] H. Jégou, M. Douze, C. Schmid and P. Perez, “Aggregating Local Descriptors into a Compact Image Representation,” in *Computer Vision & Pattern Recognition*, San Francisco, CA, USA, June, 2010.

[34] F. Pereira, E. A. da Silva and G. Lafruit, Plenoptic Imaging: Representation and Processing, Academic press library in signal processing, 2014.

[35] S. J. Gortler, R. Grzeszczuk, R. Szeliski and M. F.Owen, “The Lumigraph,” Microsoft Research, 1996.

[36] “Cgtrader,” [Online]. Available: https://www.cgtrader.com/3d-models/animals/mammal/monkey-base-mesh. [Accessed April 2016].

[37] “Dense Modelling,” [Online]. Available: http://grail.cs.washington.edu/rome/dense.html. [Accessed April 2016].

[38] J. Goldman, “Lytro Light-Field Camera Review,” [Online]. Available: http://www.cnet.com/products/lytro-light-field-camera/. [Accessed May 2016].

[39] T. Simotine, “MIT Techonology review,” [Online]. Available: http://www2.technologyreview.com/news/427671/light-field-photography/. [Accessed May 2016].

[40] “Bubl Camera Official Website,” [Online]. Available: https://www.bublcam.com/. [Accessed May 2016].

[41] “Panono Official Website,” [Online]. Available: https://www.panono.com/home. [Accessed May 2016].

[42] “Inspect,” [Online]. Available: http://www.inspect-online.com/en/produkte/vision/new-industrial-grade-3d-tof-cameras. [Accessed May 2016].

[43] D. Scharstein and R. Szeliski, “A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms,” *International Journal of Computer Vision,* pp. 7-42, April 2002.

[44] M. Karpushin, G. Valenzise and F. Dufaux, “Improving Distinctiveness of BRISK Features Using Depth Maps,” in *International Conference on Image Procesing*, Quebec City, QC, Canada, September, 2015.

[45] E. Nascimento, G. Oliveira, A. Vieira and M. Campos, “On the Development of a Robust, Fast and Lightweight Keypoint Descriptor,” *Neurocomputing,* vol. 120, pp. 141-155, November November, 2013.

[46] M. Karpushin, G. Valenzise and F. Dufaux, “Local Visual Features Extraction From Texture+Depth Content Based on Depth Image Analysis,” in *International Conference on Image Processing*, Paris, France, October, 2014.

[47] K. Mikolajczyk and C. Schmid, “An Affine Invariant Interest Point Detector,” in *European Conference on Computer Vision*, Copenhagen, Denmark, May, 2002.

[48] K. Mikolajczyk, T. Tuytelaars, C. Schmid, . Z. A., J. Matas, F. Schaffalitzky, T. Kadir and L. Van Gool, “A Comparison of Affine Region Detectors,” *International Journal of Computer Vision,* vol. 65, pp. 43-72, November November, 2005.

[49] “CTurin180 dataset,” [Online]. Available: https://pacific.tilab.com/www/datasets/. [Accessed January 2017].

[50] “The Oxford Buildings Dataset,” [Online]. Available: http://www.robots.ox.ac.uk/~vgg/data/oxbuildings/. [Accessed November 2016].

[51] “Zurich Building Image Dataset,” [Online]. Available: http://www.vision.ee.ethz.ch/showroom/zubud/. [Accessed December 2016].

[52] “The Stanford Mobile Visual Search Dataset,” [Online]. Available: https://purl.stanford.edu/ph459zk5920. [Accessed January 2017].

[53] “San Francisco Landmark Dataset,” [Online]. Available: https://purl.stanford.edu/vn158kj2087. [Accessed December 2016].

[54] N. Afonso, M. Vetterli and A. Ghasemi, “LCAV-31: A Dataset for Light field Object Recognition,” in *Computational Imaging XII*, San Francisco, California, USA, 2013.

[55] “Light Field ToolBox v0.4,” [Online]. Available: https://www.mathworks.com/matlabcentral/fileexchange/49683-light-field-toolbox-v0-4?requestedDomain=www.mathworks.com. [Accessed November 2016].

[56] “OpenCV,” [Online]. Available: http://opencv.org/downloads.html. [Accessed November 2016].

[57] “Consistent Depth Estimation in a 4D Light Field,” [Online]. Available: https://hci.iwr.uni-heidelberg.de/hci/softwares/light_field_analysis/depth.

[58] “Hough Line Transform,” [Online]. Available: http://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_houghlines/py_houghlines.html. [Accessed March 2017].

[59] “Canny Edge Detection,” [Online]. Available: http://docs.opencv.org/trunk/da/d22/tutorial_py_canny.html. [Accessed March 2017].

[60] “Sci-kit image,” [Online]. Available: http://scikit-image.org/. [Accessed March 2017].

[61] “Affine Covariant Features Dataset,” [Online]. Available: http://www.robots.ox.ac.uk/~vgg/research/affine/. [Accessed March 2017].

[62] R. Gerhard, “Homography,” [Online]. Available: http://people.scs.carleton.ca/~c_shu/Courses/comp4900d/notes/homography.pdf. [Accessed March 2017].

[63] K. Mikolajczyk and C. Schmid, “Scale & Affine Invariant Interest Point Detectors,” *International Journal of Computer Vision,* pp. 63-86, Jan 2004.

[64] K. Mikolajczyk and C. Schmid, “A Performance Evaluation of Local Descriptors,” in *Computer Vision and Pattern Recognition*, Madison, WI, USA, 2003.