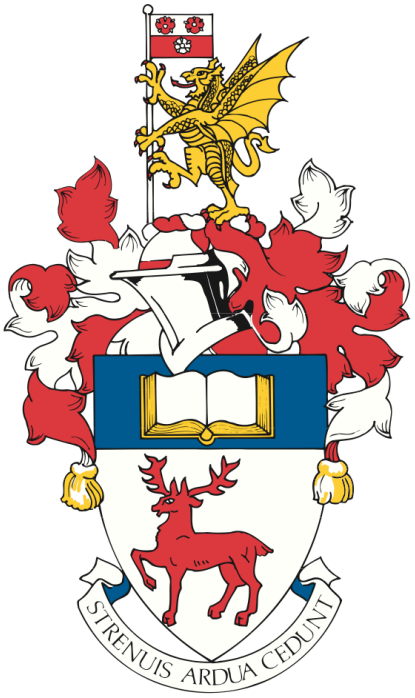
UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science



Analysis of Symmetric Patterns in Images

by

Jaime Lomelí Rodríguez

Jury

Dr. Krystian Mikolajczyk

Dr. Jonathon Hare

Supervisor

Prof. Mark S. Nixon

Thesis for the degree of Doctor of Philosophy

November 2017

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING
Electronics and Computer Science

Doctor of Philosophy

ANALYSIS OF SYMMETRIC PATTERNS IN IMAGES

by Jaime Lomelí Rodríguez

The computational detection of symmetry is of large importance not only to model human vision, but to leverage the structural information it carries. Due to the complexity of the problem, researchers tackle automated symmetry detection as a search rather than a model-based approach in most of the cases.

The detection of symmetry has found several applications including segmentation, gait analysis, interest-point detection, saliency models and human pose tracking amongst others. Despite the variety of approaches, the detection of symmetry is still an open area of research. In this thesis, symmetry perception and detection algorithms are directly related to *Marr's stages of vision*.

Feature-based symmetry detection methods tend to over-perform other approaches, nevertheless, they are computationally demanding. These are reliant on the presence of matched pairs of features, therefore they benefit from the abundance of such points; this implies that a trade-off between performance and computation time must be found. Here, the detection of large sets of features and the computation time for feature based symmetry detection algorithms are addressed.

A new procedure for feature based symmetry detection is introduced. We make use of the efficient binary descriptors, allowing for a drastic reduction on the computation times. Moreover, we use a density-based approach to detect axes of symmetry in the parameter space; this augments resolution and provides more flexibility for the detection.

Two new feature detection methods are presented. The Locally Contrasting Keypoints (LOCKY) is a novel blob-detector aimed at reducing computation times. The detection is achieved with an innovative non-deterministic low-level operator called the Brightness Clustering Transform (BCT). The BCT can be thought as a coarse-to-fine search through scale spaces for the true derivative of the image; it also mimics trans-saccadic perception of human vision. LOCKY shows good robustness to image transformations included in the Oxford affine-covariant regions dataset, and is amongst the fastest affine-covariant feature detectors.

Unsupervised representation learning enables computers to learn visual cues from unlabelled data, obviating the need for hand-crafted feature models. The second feature detector is a novel technique to find rotation-invariant structures using an unsupervised representation learning strategy. This is accomplished by mapping image-patches into a rotation-invariant space built with the Bessel-Fourier moments. We analyse this space in terms of the symmetry of features and categorise them into three groups, non-symmetric, symmetric and composite-symmetric. Feature-maps are created by comparing patches in an image against the feature models, non-maxima suppression is performed afterwards. The detected features show very competitive results on repeatability tests compared with state of the art detectors. Moreover our approach can recognise multiple structures therefore, the user can substantially increase the amount of detected features in a controlled manner.

Using the methods presented in this thesis, symmetry detection computation times are drastically reduced by approximately ten-fold compared to other state of the art approaches. The use unsupervised learning for the rotation-invariant detection of structures, also yields an improvement on the symmetry detection performance measured against the state of the art.

Contents

Declaration of Authorship xi

Acknowledgements xiii

Nomenclature xv

Abbreviations xix

1 Introduction 1
- 1.1 Context . . . 1
  - 1.1.1 Three stages of vision . . . 5
- 1.2 Contributions . . . 8
- 1.3 Publications . . . 9
- 1.4 Outline . . . 10

2 Related work 11
- 2.1 Feature based symmetry detection . . . 14
  - 2.1.1 Feature detection . . . 15
  - 2.1.2 Feature description . . . 16
  - 2.1.3 Mirror matching . . . 17
  - 2.1.4 Parameter space . . . 18
- 2.2 Conclusion . . . 19

3 A fast approach for feature-based symmetry detection 21
- 3.1 Feature detection . . . 21
- 3.2 Mirror binary descriptors . . . 21
- 3.3 Parameter space . . . 23
  - 3.3.1 Density-based symmetry axes detection . . . 24
- 3.4 Optimal axes . . . 26
- 3.5 Conclusion . . . 27

4 Locally contrasting keypoints 29
- 4.1 The brightness clustering transform . . . 30
  - 4.1.1 Blob detection . . . 30
  - 4.1.2 Ridge detection . . . 34
  - 4.1.3 Operator invariance . . . 34
- 4.2 Locally contrasting keypoints . . . 35
  - 4.2.1 Time complexity analysis . . . 36
- 4.3 Detection results . . . 37
- 4.4 Conclusion . . . 42

5 Spectral analysis of image-patches 43
- 5.1 Image Moments . . . 45
  - 5.1.1 Geometric moments . . . 46
  - 5.1.2 Central moments and moment invariants . . . 47

5.1.3 Complex moments . . . 48
5.1.4 Bessel-Fourier moments . . . 48
5.1.5 Other image moment functions . . . 51
5.2 The symmetry of features . . . 51
5.2.1 Non-periodic 2D symmetries . . . 52
5.2.2 Phase congruency in cylindrical coordinates . . . 53
5.2.3 Creating symmetry . . . 54
5.3 Unsupervised learning to find structures in a rotation invariant space. . . . 57
5.3.1 A rotation-invariant space . . . 57
5.3.2 An approach to detect low-level symmetry . . . 58
5.3.3 Manual selection of feature models . . . 59
5.3.4 Learning repetitive structures . . . 62
5.4 Feature detection . . . 63
5.4.1 Time complexity analysis . . . 66
5.5 Results . . . 67
5.6 Conclusion . . . 70

6 Symmetry detection results **71**
6.1 LOCKY . . . 71
6.2 Unsupervised learning to find structures in a rotation-invariant space . . . 74
6.3 Comparison . . . 77
6.4 Results . . . 82

7 Conclusions and future work **87**
7.1 Conclusions . . . 87
7.2 Future work . . . 88

References **91**

List of Figures

1.1 Examples of mirror-symmetry. . . . 2
1.2 Examples of glide-reflection symmetry. . . . 2
1.3 In the words Hermann Weyl... . . . 3
1.4 Deviation from exact symmetry. . . . 3
1.5 The importance of locality in computer vision. . . . 4
1.6 Global and local symmetries. . . . 4

2.1 The medial axis of the letter 'B'. . . . 13
2.2 Modern skeletonization algorithms. . . . 14
2.3 Different binary descriptor sampling patterns. . . . 17
2.4 Matching mirror features. . . . 18
2.5 The Hough transform of a line. . . . 18
2.6 Different lines in the parameter-space. . . . 19
2.7 Loy and Eklundh's process for symmetry detection. . . . 20

3.1 Flowchart for feature based symmetry detection. . . . 21
3.2 Symmetric BRISK pattern. . . . 22
3.3 Symmetric BRISK descriptor arrangement. . . . 23
3.4 The symmetry parameter-space. . . . 23
3.5 Straight and curved axes of symmetry in the parameter-space. . . . 24
3.6 Multiple axes detection example. . . . 25

4.1 LOCKY and LOCKY-S blobs detected on an image of daisies. . . . 29
4.2 A squared vote on the left and a rectangular vote on the right. . . . 31
4.3 Gradient Haar-like wavelets. . . . 32
4.4 LOCKY and LOCKY-S features. . . . 34
4.5 BCT for ridge detection. . . . 34
4.6 The process for extracting the LOCKY features. . . . 36
4.7 Images in the Oxford affine-covariant regions dataset [75]. . . . 37
4.8 LOCKY-S features in yellow and MSER regions in green. . . . 39
4.9 Repeatability results of LOCKY features. . . . 40
4.9 Continued. . . . 41

5.1 Edge-like structures detected with rotation-invariance. . . . 43
5.2 Bessel functions of the first kind for $m = 0, 1, 2, 3, 4$. . . . 49
5.3 Real part of the kernels of the BF moments. . . . 51
5.4 Phase-congruency examples. . . . 53
5.5 BF moment kernels creating symmetry. . . . 54
5.6 Synthetically generated symmetry. . . . 56
5.7 Detecting symmetric image-patches using the presented framework. . . . 59
5.8 The flow for mapping the image into a rotation-invariant space. . . . 60
5.9 A rank-3 approximation of the normalised vectors. . . . 62
5.10 The nine closest patches to each of the learnt centroids. . . . 64
5.11 The distance function $D_c(x, y)$. . . . 65

5.12 Using the learnt centroids we create feature maps $D_c(x, y)$. 66
5.13 The 16×16 Bessel-Fourier kernels used for the detection of features. . . . 67
5.14 The first and sixth images of the boat, graffiti and semper sequences. . . . 67
5.15 Results of the repeatability test presented in [75]. 68
5.16 Every feature class is measured for repeatability. 69

6.1 Implementation of LOCKY for symmetry detection. 71
6.2 Symmetry results using LOCKY. 72
6.3 Symmetry detection fail cases using LOCKY. 73
6.4 Use of learnt structures for symmetry detection. 74
6.5 Symmetry results using the learnt structures. 76
6.6 Symmetry detection fail cases using the learnt structures. 77
6.7 Comparison between the matching time of traditional descriptors and binary descriptors. 78
6.8 Ground truth comparison. 80
6.9 Comparison to [52]. 81
6.10 Straight and curved reflection axes detection using the learnt structures as the detection strategy. 82
6.11 Straight and curved glide-reflection axes detection using the learnt structures as the detection strategy. 83
6.12 Detection results on wallpaper patterns using the learnt structures as the detection strategy. 84
6.13 Miscellaneous fail cases using the learnt structures as the detection strategy. 85

List of Tables

4.1 The average computation time for a set of 127 images, expressed in milliseconds. The images were converted to grayscale with 1024×768 pixels. The dispersion index shows how scattered the features are across the image (calculated with equation 4.11 averaged over the same set of 127 images, the lower the index the more evenly-distributed). 38

5.1 Non-periodic symmetries of image-patches as presented in [34]. 52
5.2 Existence and phase conditions for image moment functions to create symmetric patterns. 55

6.1 Feature detection times on rescaled images. 77
6.2 Results comparing our approach to those showed by Lee and Liu in [52]. . 79

Declaration of Authorship

I, Jaime Lomelí Rodríguez , declare that the thesis entitled *Analysis of Symmetric Patterns in Images* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: [61], [62] and [63]

Signed:.....

Date:.....

Acknowledgements

I express my gratitude to Prof. Mark, for helping make my PhD an enjoyable journey. I also want to thank my colleagues for all the great experiences we have lived. To the Light Opera Society who became a second family during these years. To all the people who gifted me with so many happy memories and their friendship. And most importantly, to my family. My parents and my sisters are my pillars of support.

Thank you.

Nomenclature

Chapter 1

S	symmetric mapping
$\mathbf{p}$	an arbitrary point

Chapter 2

$f(t)$	generic function of time
$\overline{f(t)}$	complex conjugate of $f(t)$
τ	autocorrelation time lag
$F(\upsilon)$	Fourier transform of f
$FT\{f\}$	Fourier transform operator *i.e.* $FT\{f(t)\} = F(\upsilon)$
υ	Frequency domain independent variable
r	Hough transform, distance of a point to the origin
ϕ	Hough transform, deviation angle from horizon

Chapter 3

i, j, p, l	indexing variables
$\mathbf{p}_i$	i^{th} detected feature
(x_i, y_i)	spatial location of $\mathbf{p}_i$
s_i	scale of $\mathbf{p}_i$
ϕ_i	orientation of $\mathbf{p}_i$
$\mathbf{b}_i$	binary descriptor of $\mathbf{p}_i$
b_p	p^{th} comparisson result of a binary descriptor
$\Delta(\mathbf{b}_i, \mathbf{b}_j)$	Hamming distance between $\mathbf{b}_i$ and $\mathbf{b}_j$
$\mathbf{p}_{ij}$	matching pair batween $\mathbf{p}_i$ and $\mathbf{p}_j$
$BRISK_{thresh}$	Hamming distance threshold
(x_c, y_c)	coordinates of the centre of the image
(r_{ij}, ϕ_{axis})	coordinates of $\mathbf{p}_{ij}$ in the parameter space
ϵ	DBSCAN: density factor
$minPts$	DBSCAN: minimum number of density reachable points
L	DBSCAN: number of detected clusters (detected axes)
$\mathbf{P}$	the set of all successful matches $\mathbf{p}_{ij}$
$\mathbf{O}$	subset of $\mathbf{P}$ containing outliers
$\mathbf{R}_l$	l^{th} subset of $\mathbf{P}$ with elements in a cluster
A	cardinality of $\mathbf{R}_l$
d	degree of a polynomial regression
σ	rotation angle for polynomial regression
$\mathbf{y}$	vector of y values of the middle points of $\mathbf{p}_{ij} \in \mathbf{R}_l$

$\mathbf{X}$	Vandermonde matrix of x values of the middle points of $\mathbf{p}_{ij} \in \mathbf{R}_l$
$\mathbf{q}$	coefficients of the polynomial of degree d that best describes the data
$\mathbf{z}$	vector of errors between the data and the estimated model $\mathbf{q}$

Chapter 4

t, i, m, n	indexing variables
$f(x, y)$	image function with cartesian coordinates
$s(x, y)$	cumulative row sum of the image function
$ss(x, y)$	integral image
V	BCT: total number of votes
I_t	BCT: t^{th} iteration region
ι_i	BCT: subdivisions of I_t
u, v	BCT: width and height exponent settings
(x_f, y_f)	BCT: final ι_i location
$width_f, height_f$	BCT: final ι_i width and height
loc	BCT: location of the vote in the accumulator matrix
$rangeMin$	BCT: minimum value for u and v
$rangeMax$	BCT: maximum value for u and v
$g(x, y)$	illumination function
(x_m, y_m)	pixel location vector
M	total number of pixels in a binary connected component
$()^T$	vector transposition operator
$\mathbf{Q}$	sample covariance matrix
B	the number of non-overlaping bins
P	total number of detected features
X	total number of pixels in the image
$\mathbf{p_i}$	location vector of the i^{th} feature
Γ_b	the region of a bin
C_b	count of features in the region Γ_b
D	the disperssion index

Chapter 5

i	the complex variable
I	region where the image is defined
p, q	geometric moment order
ζ_p	p^{th} order moment of a punctual particle
$\zeta_{p,q}$	$(p+q)^{th}$ order geometric moment
$f(x)$	generic function
$f(x, y)$	image function (Cartesian)
(x, y)	Cartesian coordinates
$F(\upsilon)$	Fourier transform of $f(x)$

$F(\upsilon, \nu)$	Fourier transform of $f(x, y)$ (Cartesian)
(υ, ν)	frequency domain coordinates (Cartesian)
$\mu_{p,q}$	$(p+q)^{th}$ order central moment
$\eta_{p,q}$	$(p+q)^{th}$ order scale-normalised central moment
$\psi_{m,n}$	$(m+n)^{th}$ order complex radial moment
$\Upsilon_{m,n}(r)$	radial moment generating function
$f(r, \theta)$	image function (cylindrical)
(r, θ)	cylindrical coordinates
$F(\rho, \gamma)$	Fourier transform of $f(r, \theta)$ (cylindrical)
(ρ, γ)	frequency domain coordinates (cylindrical)
ϕ	angle lag
$J_m(\rho)$	m^{th} order Bessel function of the first kind
$\lambda_{m,n}$	n^{th} zero of the m^{th} order Bessel function of the first kind
$H_m(\rho)$	Hankel transform of order m
$B_{m,n}$	$(m+n)^{th}$ order Bessel-Fourier moment generating function
$R_{m,n}(r)$	$(m+n)^{th}$ order Zernike moment generating function
$S_{s,k}$	s^{th} non-periodic symmetry as presented in [34]
$f(t), g(t)$	generic functions
k	number of repetitions of a given symmetry
M, N	maximum number of extracted $B_{m,n}$ moments
$\mathbf{d}_x$	$M \times N$-dimensional vector of complex magnitudes of $B_{m,n}$
$\mathbf{d}_{x,y}$	$\mathbf{d}_x$ of the pixel located in (x, y)
E	energy of a signal
E_d	energy of the vector $\mathbf{d}_x$
E_{thresh}	energy threshold
C	number of feature models to learn
ω_c	c^{th} feature model (centroid)
$\langle , \rangle$	the dot product operator
$D_c(x, y)$	c^{th} feature map
τ	similarity tolerance
D_{thresh}	non-maxima supression threshold for the feature maps
X	total number of pixels in the image

Abbreviations

LOCKY	Locally Contrasting Keypoints
CNN	Convolutional Neural Network
DST	Discrete Symmetry Transform
GST	Generalised Symmetry Transform
MUST	Multi-scale Symmetry Transform
FT	Fourier Transform
LoG	Laplacian of Gaussian
SUSAN	Smallest Univalue Segment Assimilating Nucleus
FAST	Features from Accelerated Segment Test
AGAST	Adaptive Generic Accelerated Segment Test
CenSurE	Centre Surround Extrema
MSER	Maximally Stable Extremal Regions
SIFT	Scale Invariant Feature Transform
SURF	Speed-Up Robust Features
HOG	Histogram of Oriented Gradients
BRIEF	Binary Robust Independent Elementary Features
ORB	Oriented FAST and Rotated BRIEF
BRISK	Binary Robust Invariant Scalable Keypoints
FREAK	Fast Retina Keypoints
DBSCAN	Density Based Spatial Clustering of Applications with Noise
BCT	Brightness Clustering Transform
BF	Bessel-Fourier
PDF	Probability Density Function

Chapter 1

Introduction

1.1 Context

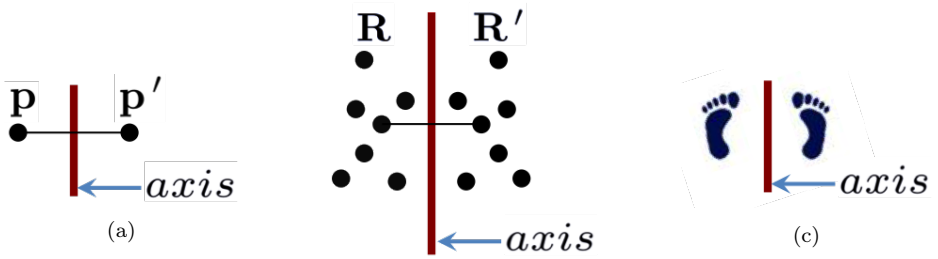
The perception of *symmetry* has long been studied. Since ancient times, symmetry has been deeply related to beauty. Along the way, philosophers and researchers have found that symmetry means much more than visual appeal. Studies have proved that symmetries are building-blocks of structured patterns, therefore, symmetry is an important factor in the perception of shape. This would imply that symmetry is vital in more complex visual tasks such as object recognition and segmentation.

Humans' remarkable ability to discern symmetrical patterns in cluttered scenarios has intrigued neuroscientists for many years. Researchers suggest that the perception of mirror symmetry arises from symmetric neuronal interconnections between both hemispheres of the brain [80]. It is also recognised that contour grouping could be assisted by the presence of symmetry [90]. Treder gives a review on the perception of symmetry, analysing how factors such as rotation or noise affect humans' ability to recognise its existence [97].

Symmetry has also attracted the attention of mathematicians and, more recently, computer scientists. For instance, eyes can be landmarked and tracked using their round shape as a cue. Eye tracking systems have several applications for finding points of high saliency or attention [8]. Car license plates can be automatically detected by exploiting their rectangular symmetric shape [44]. Automatic Number Plate Recognition finds applications like law-enforcement or location tracking.

Researchers also use symmetry to detect salient regions on images [46, 33]. Symmetry can also be used to analyse gait [38], such methods help to recognise people in low-resolution recordings or when the individual's face is not visible. Other applications for the automated detection of symmetry include interest-point detection, grouping, human pose tracking or image segmentation to name a few [67, 89, 37, 45, 82, 84].

Symmetry Lets define symmetry in a geometrical way. Consider the transformation S that maps a point $\mathbf{p}$ into a corresponding point $\mathbf{p}'$ *i.e.* $S : \mathbf{p} \mapsto \mathbf{p}'$. And its inverse



(a) (b) (c)

Figure 1.1: Mirror-symmetry. A reflection across the axis keeps a figure unchanged.

transformation S' which maps $\mathbf{p}'$ back into $\mathbf{p}$ *i.e.* $S' : \mathbf{p}' \mapsto \mathbf{p}$. The successive application of both these transformations to a point is an operation that preserves the structure of space making it an *automorphism i.e.* $\mathbf{p}$ remains unchanged.

The transformation S that takes $\mathbf{p}$ to the 'opposite' side of a straight line is a *reflection*. This line is known as the *axis of symmetry*, $\mathbf{p}$ and $\mathbf{p}'$ form a mirror-symmetric pair (figure 1.1a). In a similar manner, if S reflects a set of points $\mathbf{R} = \{\mathbf{p}_i \,|\, i = 0, 1, ..., A\}$ across the same axis into $\mathbf{R}' = \{\mathbf{p}'_i \,|\, i = 0, 1, ..., A\}$, the union $\mathbf{R} \cup \mathbf{R}'$ forms a mirror-symmetric group (figure 1.1b). An image (or an object) is mirror-symmetric if there is a reflection S across an axis that leaves it unchanged (figure 1.1c).

In addition to the reflection, there are other transformations that preserve some objects across the operation. An object that can be periodically rotated around a point, while remaining unchanged, is known to be rotational-symmetric. An example of this is shown in figure 1.6a.

In conclusion, a *symmetric object* is one that is invariant to some geometric transformation other than the identity transformation, and the transformations that leave such objects unchanged are known as *symmetries e.g.* rotation, translation, reflection, scaling.

Figure 1.2: Examples of glide-reflection symmetry. A reflection plus a translation across the axis generates glide-reflection symmetric figures.

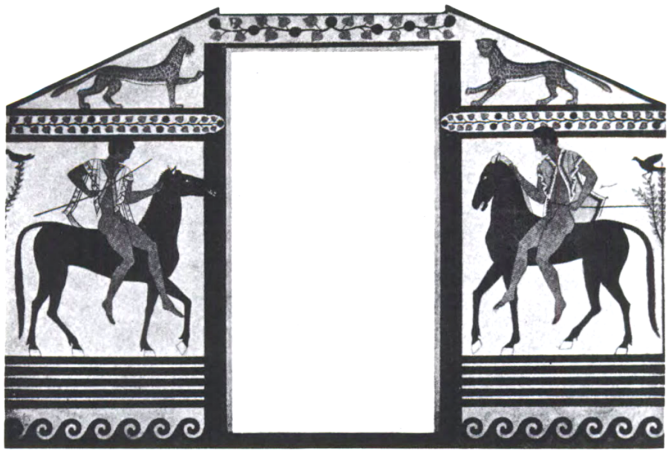


Figure 1.3: In the words of Hermann Weyl, "Seldom is asymmetry merely the absence of symmetry". Asymmetry is a deviation from exact symmetry.

Symmetries can also be a combination of transformations. For instance, a reflection and a translation across its axis is known as a *glide-reflection symmetry*; figure 1.2 illustrates this.

Weyl defines symmetry as something like well-proportioned or well- balanced [106]. Symmetry becomes a more complex matter when objects begin to deviate from the perfect symmetry described above. "Seldom is asymmetry merely the absence of symmetry. Even in asymmetric designs one feels symmetry as the norm from which one deviates under the influence of forces of non-formal character," says Weyl when talking about the inclination of occidental art to break strict symmetry, exemplifying with the Etruscan painting of riders shown in figure 1.3.

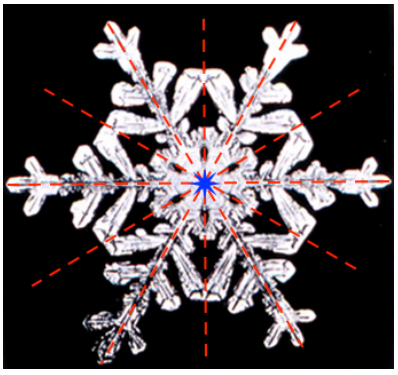
Structural artefacts or deformations break exact symmetry, this phenomenon is frequently found in real-world images. An example of this is pictured on the left side of figure 1.4, where a glide-symmetric pattern is constructed from irregular tiles. Curved axes of symmetry can also be a deviation from exact symmetry, as points are not reflected across a straight line; this is shown on the right side of figure 1.4.

Figure 1.4: Deviation from exact glide-symmetry in an image of a mosaic pattern (left). A curved axis of symmetry is also a deviation from exact symmetry.

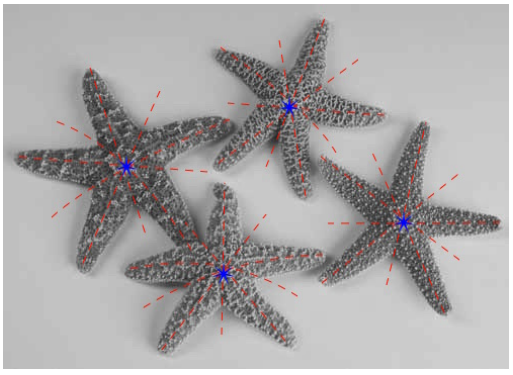
Locality In practice, images are bounded to the extent of a pixel matrix. Generally, scenes are formed of simpler structural components and may contain an infinitely large amount of combinations of them. For example, if we analyse the image in figure 1.5 as a whole, its description would not be *a viaduct, mountains, a train, etc.*; instead, it would be this specific collection of intensities. Changing the position of any of the objects in the image would change the intensities in some regions, thus the description of the entire image would change.

Figure 1.5: The importance of locality in computer vision.

On the other hand, if we consider the image to be composed of smaller *local structural elements* or *objects* that are finite in extent, one can describe the image in a more flexible manner. Locality refers to *finite extensions of images* or *image-patches*. The definition of locality triggers the concept of scale, as image-patches of different sizes may present the same structural characteristics.



(a) A snowflake.



(b) Starfish.

Figure 1.6: The image of a snowflake with global symmetry. Starfish are non-rigid objects, here each represents a local symmetry. Axes of mirror symmetry in red and, centres of rotational symmetry in blue.

We refer as *local symmetry* to the symmetries that are present at the local level and, *global symmetry* to the symmetries that span the entire space of an image. Figure 1.6 shows both local and global types of symmetry; the snowflake spans the whole image, whilst the starfish are small portions of the image.

1.1.1 Three stages of vision

According to David Marr, visual processing is performed at three different stages. On the top level there is a 3D model of the world that surrounds us. Breaking down this 3D model, there are textures and planes that constitute the 2.5D sketch. The bottom stage of visual perception is the primal sketch: *"The primal sketch consists of primitives of the same general kind at different scales... but the primitives can be defined from an image in a variety of ways, from the very concrete (a black ink mark) to the very abstract (a cloud of dots)"* [71]. For ease of understanding and generalisation, we rename Marr's stages of vision:

- Low-level sketch: the primal sketch of vision. This encompasses low levels of image processing like filtering and feature detection.
- Mid-level sketch: the 2.5D sketch. Shape and feature grouping, primal segmentation.
- High-level sketch: the 3D model. We know a model of the object and we can understand it. We know what we are looking at and are able to retrieve more information due to the known model.

Simple or complex, all visual cues may show any type of symmetry, or none. As objects deviate from exact symmetry, the task of finding corresponding symmetric pairs of points goes from row perception to grouping, requiring higher levels of cognition [97]. Perception of symmetry can then be related to Marr's stages of vision.

Low-level sketch Image-patches that present certain structural characteristics are known as *keypoints* or *features*, and the process of finding them is known as *feature detection*. For instance, it is possible to measure how similar an image-patch is to an 'ideal corner'; image-patches that are similar to such model are considered as 'corners'.

Feature detection is related to the lower levels of vision. It must either provide a measure (covariance) or return the same result (invariance) after certain image transformations including scale, rotation and noise amongst others. The term *covariance* was suggested by Tuytelaars and Mikolajczyk [99], it is to be used when features change covariantly with the image transformation. On the other hand, *invariance* refers to the fact that a measure is not affected by the image transformation. In chapter 4, a new non-deterministic feature detection method named the *Locally Contrasting Keypoints* detector (LOCKY) is presented. LOCKY shows good robustness against image transformations, and is amongst the fastest affine-covariant feature detectors.

Humans extract information from scenes as a series of snapshots known as fixations, the rapid eye movements between fixations are known as saccades. There is a temporal

memory that integrates fixations to form a single percept through various saccades [43]. This process is called *trans-saccadic integration*. LOCKY is based in a non-deterministic algorithm that employs a voting scheme. Each vote is randomly initialised, then it extracts information from the region being attended resembling eye fixations. When the next vote is initialised, the algorithm changes the location of attention, this is a saccade.

Computer vision has taken great advantage from the development of powerful computers. More complex and computationally demanding calculations are now possible to process thus, computers can interpret large amounts of data to find repetitive patterns of different forms. The field of Convolutional Neural Networks (CNN's) has benefited from this fact, and applications using such methods seem to constantly over-perform those that do not.

Many aspects of the CNN framework remain open for discussion, one of them is their relation to the detection of features. Classic feature detection algorithms find specific hand-crafted structures that can be categorised into general groups, edges, ridges, corners, blobs or regions, depending on the structure they represent. On the other hand CNN training algorithms, better known as *representation learning*, take large sets of images to find patterns that are persistent throughout the data. The result of this training process is a set of filters that are stacked on layers as a deep network, increasingly more complex patterns are found on each layer. This allows for the extraction of statistically more consistent features.

A feature can be any visual cue, namely, a corner, an edge, a 'black ink mark' or a 'cloud of dots'. In chapter 5, a novel feature detection technique is introduced that fins structures with rotation-invariance. This detector exploits the representation learning strategy to find repetitive features of any kind. Features are analysed in terms of their symmetry characteristics and categorised into non-symmetric, symmetric and composite-symmetric groups. Feature detection is a search-space reduction in which a large amount of information is discarded. Traditional feature detectors introduce a trade-off between feature density and distinctiveness. Our approach circumvents this issue by allowing the detection of any particular structure.

In a similar fashion, exact (or nearly exact) symmetry can be found with low levels of attention. In real-world images, exact symmetry is far more often found in small regions as symmetric image-patches than as global structures. The low-levels of symmetry can be detected using correlation, spectral or other type of low-level image analysis without previous segmentation.

Mid-level sketch Groups of features form more complex structures that conform the mid-level sketch. These structures can be recognised as textures or other patterns. Mid-levels of symmetry are features arranged in an approximately symmetrical manner, as objects show some degree of deviation from exact symmetry. When there is enough information on the image to determine whether an object is symmetric or not. This level

of cognition allows the detection of symmetry in unseen objects that do not show perfect left-right balance. Mid-level symmetry perception includes the detection of symmetries in occluded or skewed objects.

The 2D projection of a deformable symmetric object does not show perfect balance, since the deformed parts modify curvature and brightness. There is enough evidence suggesting that the object could be symmetric after a certain transformation. Nonetheless, our knowledge of deformable objects intervenes and suggests that symmetric objects can modify certain characteristics while retaining an arrangement of symmetric (corresponding) features.

The computational detection of mid-level symmetric structures requires the search for corresponding regions in an image. The more complex nature of this level of symmetry introduces problems like occlusions and noise. To deal with these problems, researchers make use of feature detection, energy optimization or machine learning techniques amongst others.

The detection of symmetry at this level is usually a computationally expensive process. The feature based approaches have gained attention as they tend to over-perform other algorithms, the idea was first introduced by Loy and Eklundh [66]. These methods rely on the detection of features in images, these features are then described and compared against each other to find correspondences. Geometrical information extracted from these correspondences highlights probable axes of symmetry in the image. The abundance of detected features benefits the process, this implies that a trade-off between performance and computation time must be found.

This thesis explains a new set of techniques to reduce the computation time of feature based symmetry detection. This is achieved by modifying or replacing some of the stages involved in the process. Computation times are drastically shortened taking an average of 0.59 seconds to process an image. Detection results for single and multiple, straight and curved, reflection and glide-reflection symmetries are similar to the current state of the art.

High-level sketch The high-level sketch of perception, involves deeper levels of knowledge and processing of scenes. When the 2D projection of a 3D object does not contain any information on the symmetry of an object at all, high-level cognition of symmetry allows us to recognise symmetrical objects. For example, if a face is rotated by 90° in the depth plane no signs of symmetry are present in the image, but our knowledge and model of faces indicate the presence of symmetry.

Also, symmetrical arrangements of objects of different characteristics can be thought as symmetry. Symmetrical arrangements of objects are detected after a longer process of reasoning and recognition. Liu *et al.* suggest that the symmetric arrangement of a square

and a deforming square into a triangle is an ambiguity in mirror symmetry, real world images of this shape ambiguity can be the mirror arrangement of two objects of the same kind with different shape or characteristics [59].

No algorithm is recognized that uses a high-level of cognition for symmetry detection. The reason for this could be that, generally, symmetry detection serves as a tool rather than an objective in computer vision.

From symmetric primitives in the Low-level sketch to symmetrical 3D structures in the High-level sketch, symmetry is embedded in every stage of vision. The computational detection of symmetry is of great importance not only to model human vision, but to leverage the structural information symmetry carries. Moreover, understanding the structure of features could bridge the gap of knowledge that relates low-level image processing and the higher order computer vision operations.

1.2 Contributions

This thesis addresses the slow computation times of the feature based symmetry detection approach, to achieve this, a set of improvements to the method are introduced. These improvements reduce computation times and augment resolution to maintain state of the art performance.

Moreover, two new feature detectors are presented. The first uses a non-deterministic approach to detect blobs and measure affinity. The second method relies on an in-depth analysis of the relation between features and low-level symmetry. Such analysis conducts to a generalised categorisation of visual primitives into non-symmetric, symmetric and composite-symmetric groups, strengthening comprehension on the structure of features.

This thesis expands both, symmetry and feature detection literature in theoretical and practical contexts with methods that compare with the state of the art. A condensed list of contributions is shown below.

- Symmetry perception in human vision appears to be processed at both conscious and sub-conscious instances [97]. We introduce a new categorisation of the perception of symmetry into low, mid and high levels of cognition, these levels can be directly related to Marr's stages of vision.
- Feature based techniques tend to over-perform other symmetry detection strategies [59]. This advantage in performance is obliterated by the time-consuming procedure for feature based detection. We address this problem by improving or replacing each of the steps on this method.

- A novel non-deterministic feature detection technique named LOCKY is presented. LOCKY is among the fastest affine-covariant approaches, showing good repeatability scores [75] and presenting the best feature dispersion across the image.
- A derivation of the generalised complex image-moments is presented. The particular case when the radial functions are the *Bessel functions of the first kind*, represents the Fourier transform in cylindrical coordinates. The coefficients of this transform are known as the *Bessel-Fourier moments* [109].
- Bessel-Fourier moments describe structural properties of image-patches. From these values, we categorise structures into three general groups, *non-symmetric, symmetric and composite-symmetric*, providing a more complete alternative to the usual corner, blob and region categorisation of features.
- A new technique to build feature-maps from a rotation-invariant space is presented. This method uses the *unsupervised representation learning* procedure to learn feature models in a rotation-invariant space. Features detected with this technique show very competitive average repeatability scores [75]. Moreover, this approach can recognise multiple structures as features thus, the user can substantially increase the amount of detected features in a controlled manner.
- Feature based symmetry detection is performed using the presented rotation-invariant representation learning approach as the feature detection stage. This technique increments the number of detected features while maintaining low computation times, taking an average of $0.59s$ to process an image in a single threaded implementation. Detection results for single and multiple, straight and curved, reflection and glide-reflection symmetries are similar to the current state of the art.

1.3 Publications

Parts of the work presented in this thesis have been published as,

- The Brightness Clustering Transform and Locally Contrasting Keypoints. In International Conference on Computer Analysis of Images and Patterns, 362-373. Springer, 2015. [61]. Best paper prize.
- An extension to the brightness clustering transform and locally contrasting keypoints. Machine Vision and Applications, 27(8), 1187-1196. Springer, 2016. [62].
- Learning Salient Structures for the Analysis of Symmetric Patterns. International Conference on Image Analysis and Recognition, 286-295. Springer, 2017. [63].
- The Symmetry of Features: Unsupervised Learning of Rotation-Invariant Structures. Submitted to IEEE Transactions on Image Processing.

1.4 Outline

The remaining of this thesis is structured as follows.

Chapter 2 shows a topical symmetry detection literature review. Algorithms for low and mid levels of symmetry detection are discussed. The flow for feature based approaches is explained in detail, and an introduction to the feature detection and description literature is also presented.

Chapter 3 explains the new process for feature based symmetry detection. In this chapter, the feature detection stage is considered as a black-box. A process for efficiently mirroring binary descriptors is introduced. Other approaches use a discretised parameter-space, a process to avoid such discretisation is described.

Chapter 4 introduces a blob detection algorithm that is easy to implement and is faster than most of the currently available feature detectors. This detector is based in a non-deterministic low-level operator named the *Brightness Clustering Transform* (BCT). The new algorithm is called *Locally Contrasting Keypoints* (LOCKY). Showing good robustness to image transformations included in the Oxford affine-covariant regions dataset, LOCKY is amongst the fastest affine-covariant feature detectors.

Chapter 5 contains a deep analysis on the structure of features. This chapter begins with an introduction to the image-moments framework. We derive the generalised complex image-moment generating function from the Fourier transform in cylindrical coordinates. Using the magnitude of complex image-moments, image-patches are mapped into a rotation-invariant space where visually similar structures are close to each other. We exploit this property to cluster large amounts of unlabelled visual cues to find repetitive model structures. A novel generic feature detection scheme that uses such model structures is presented. Features detected with this innovative method are inherently consistent, showing state of the art scores on Mikolajczyk's repeatability test [75].

Chapter 6 shows symmetry detection results using the introduced method. These, and the results obtained by other authors are compared. The computation times for different stages of the algorithm are analysed. Result images with different types of symmetry are presented.

Chapter 7 concludes the thesis and discusses new possible research directions.

Chapter 2

Related work

Symmetry detection algorithms can be directly related to the symmetry levels of humans' perception, usually requiring different approaches for their detection. Low-level symmetry relates to spectral or transformation-driven methods and can achieve feature detection; on the other hand, mid-level symmetry is related to matching, involving the previous detection of features or segmentation.

Symmetry is a recurring cue in natural and artificial objects, although perfect left-right balance is rarely found in such scenes. Imperfections break the assumption that symmetry has a binary existence *i.e.* it either exists or it does not. A number of publications treat symmetry as a continuous feature, the measure of symmetry expresses the value of shape and provides a means to compare structures [112, 111, 113, 114].

Perhaps the most straight-forward method for detecting symmetry is applying a symmetric transformation to an image and then comparing it with the original. This could be considered the brute-force approach, all the possible symmetric transforms, scales, translations and rotations have to be tested to detect all symmetries. This approach was implemented in various publications [48, 49, 17]. This solution is not only extremely computationally expensive; it also lacks robustness to noise and occlusion, also it does not provide with a good continuous measure of symmetry.

Some symmetry detection algorithms weigh the influence of brightness, gradient or local phase; a measure of symmetry is given for a pixel at a certain scale (or a linear combination of measures at different scales). The Discrete Symmetry Transform (DST) [24] and the Generalised Symmetry Transform (GST) [85], are early examples of such approach.

Loy and Zelinsky present an operator to find radially symmetric points of interest using the gradient of an image [67]. This algorithm was extended to a multi-scale and rotation invariant detector called the MUlti-scale Symmetry Transform (MUST) [45]. Sela and Levine measure symmetry by finding co-circular edges [91]. Lin and Lin present an algorithm to detect facial features using a masking scheme with gradient direction [55].

Salti *et al.* detect features with the progression of the wave equation [89]. Based on the gradient of the image, points in between symmetric contours tend to be detected. Strong responses are generated in resonating points, the resonance happens at different times in the progression providing information about the scale of the features.

Using the response of wavelet filters, Kovesi measures the *phase congruency* of composing waves [47]. In some sense this is a left-to-right similarity measure independent of brightness or contrast. Phase congruency is an alignment in the phase of sinusoidal functions around a point. For instance, the functions $sin(2t)$ and $-sin(3t)$ are congruent around $t = 0$ as both have phase equal to zero. A more in-depth discussion on this subject is found in page 53.

Griffin and Lillholm show how symmetry can be detected with derivative of Gaussian filters [35]. This approach maps the response of such filters into a parameter space that accounts for the magnitude and the angle of the filtered region. Symmetric features are robust against several image transformations. Hauagge and Snavely find symmetric points of interest using the idea of self-similarities [37].

Lee *et al.* find autocorrelation across a frieze-expansion around a point in the image to find rotational symmetry [51]. The autocorrelation of a function $f(t)$ is defined as the correlation (denoted with the $\star$ operator) of a function with itself, shown in equation 2.1.

$$(f \star f)(\tau) = \int_{-\infty}^{\infty} \overline{f(t)}\ f(t+\tau)dt. \tag{2.1}$$

$\overline{f(t)}$ is the complex conjugate of $f(t)$. Using the convolution property of the Fourier Transform (FT), this operation can be expressed as a multiplication in the frequency domain (equation 2.2).

$$FT\{f \star f\} = \overline{F(\upsilon)} \cdot F(\upsilon). \tag{2.2}$$

Here, $F(\upsilon)$ is the Fourier transform of $f(t)$. This is efficiently calculated with the discrete Fourier transform of the frieze-expansion. The condensed idea of Lee *et al.* , is to find rotational periodicity of image-patches.

The locality of symmetry provides a more relaxed definition, an axis of symmetry can be considered as an equidistant point from two or more of the boundaries of a shape. This problem is known as 'medial axis detection' or 'skeletonization'. Figure 2.1 shows the shape of the capital letter 'B' and its medial axis.

Skeletonization was first introduced by Blum as a shape description technique [16]. Given a binary shape, Blum's algorithm finds all inscribed circles that touch the boundary of the shape in at least two points. The centre of such circles are classified as part of the medial axis. The operator responds to regions in the image that show local symmetry. These local symmetries are also known as 'ribbons'. A ribbon is a shape generated by a translating geometric figure, known as 'generator', along a plane curve *i.e.* the medial axis. After some preprocessing, segmentation, edge detection or other, modern skeletonization methods find corresponding contours that create local reflection

symmetry in grayscale or colour images. These algorithms find higher levels of symmetry compared with Blum's technique.

Such methods find applications in areas like segmentation [96, 98], shape abstraction [25], object detection [93], or text-line detection in natural scenes [115]. This is achieved using a range of different approaches. For example, Levinshtein *et al.* first segments the image using super-pixels to estimate closed contours by finding region boundaries, and then group such segments [54].

Figure 2.1: The medial axis of the letter 'B'.

Widynski *et al.* tackle the problem with spatial tracking by following contours and grouping them [107]. Shen *et al.* treat skeletonization as a classification and regression problem [93]. Using a deep convolutional network their method determines whether a pixel belongs to a skeleton or not, the scale of the skeleton is approximated with additional side output layers. A purely deep neural network approach was recently introduced by Funk and Liu [32]. Their neural network generates symmetry heatmaps that mimic human labelled ground truth data.

Skeletonization constrains the detection of more intricate structures from the very definition of 'ribbon'. The skeleton shown in figure 2.1 shows how small perturbations on the objects boundaries result in undesired skeleton ramifications. Such evident problem is one of the main focus of research in medial-axis detection.

Figure 2.2 shows the results obtained by some modern skeletonization algorithms. Locality of the extracted axes is useful for object segmentation in images with clear ribbon-like structures *i.e.* the top row. However, this advantage turns into a shortcoming in symmetric structures with more intricate textures *i.e.* the middle row. Skeletonization has improved over the years, nonetheless, the long-standing issue of undesired skeleton ramification is still a limitation.

Boundaries of structures with multiple parts and the boundaries of such parts, are difficult to deal with when thinking about symmetry as ribbons. Using a more straight forward definition, feature based detectors consider a region as symmetric when it shows

multiple well-organised local visual similarities. This allows for relaxation of different assumptions. For instance, local cues (*i.e.* features) are independent from each other, providing a means to deal with occlusions. Moreover, invariance and robustness of feature detectors provide a good basis for a less constrained symmetry detection.

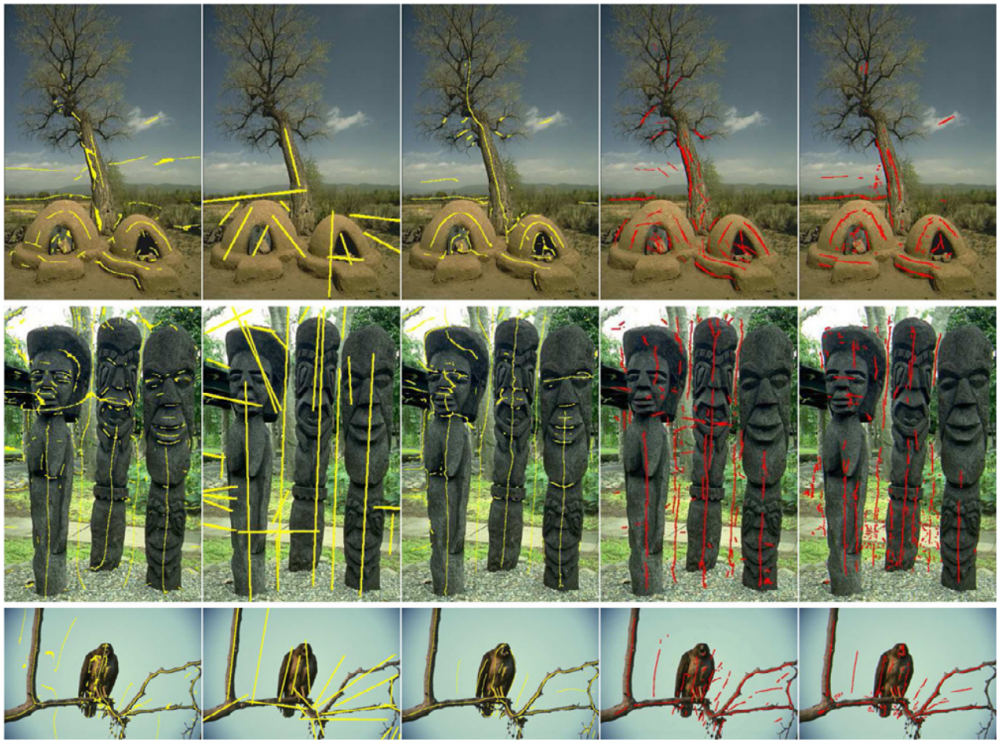


Figure 2.2: Modern skeletonization algorithms operating in real-world images. From left to right [54, 56, 98, 107, 73]. This image was taken from [73].

2.1 Feature based symmetry detection

The feature based approaches have gained attention as they tend to over-perform other algorithms in detecting symmetry. The idea was first introduced by Loy and Eklundh [66]. These methods rely on the detection of features in images, these features are then described and compared against each other to find correspondences. This reduces the search space and eliminates intermediate distortions that could disrupt the detection of symmetry. Geometrical information extracted from the correspondences highlights probable axes of symmetry in the image.

For example, Wang *et al.* present an algorithm that searches for corresponding edges to find symmetric patterns [105, 104]. The process extracts edges with the Canny detector, the edges are described using linear algebra methods. Chertok and Keller make use of several feature detectors and a spectral matching scheme to find corresponding structures [19]. Atadjanov and Lee detect contour features and use histograms of curvature to compare them to each other [6]. Other feature based approaches are presented in [52, 79].

2.1.1 Feature detection

The problem of finding features has been addressed with a wide range of solutions. Marr and Hildreth argue in the *Theory of Edge Detection* [70], that the most satisfactory operator to detect edges is the Laplacian of Gaussian (LoG) filter as edges are zero-crossings on the second derivative of a function; moreover, blobs are local-extrema when filtering with the LoG. Harris and Stephens presented a combined corner and edge detection algorithm [36]. Finding these primitives is directly related to image filtering and therefore to Fourier analysis, also, primal sketches can be directly related to the primary visual cortex (V1) of the brain.

Feature detection is one of the most basic operations in computer vision, it is a relatively fast process with timings usually varying in the range of the milliseconds in single-threaded implementations in modern computers. Many state-of-the-art applications in areas like pose estimation, image retrieval [42, 116, 58], texture description [20], keypoint prediction [64], visual content analysis [10] and scene labelling [29] to name a few, benefit from the detection of features in images.

The area of feature detection is vast, including algorithms for the detection of corners, blobs and regions. The Harris corner detector is perhaps the most well known feature detector, based on the eigenvalues of the second order moment matrix; corners can be detected with rotation invariance [36]. In need of faster algorithms other solutions have been proposed. By comparing intensities within a circular area, the SUSAN corner detector improves computation times over the Harris detector [94]. SUSAN is not very accurate for telling edges and corners apart, for this reason it is also used as an edge detector. The FAST detector compares intensities of approximately equidistant pixels from a centre *i.e.* a circle around the pixel being analysed (the AST measure); if N contiguous pixels are brighter or darker than the centre pixel (plus a threshold), this point is a potential corner [87]. To make such intensity comparisons more efficiently, FAST uses a decision tree classifier. More recently, AGAST introduced a decision tree for classifying pixels based on the AST measure much more efficiently than FAST [68]. To measure the scale of features, BRISK uses a multi-scale AGAST detector [53]. BRISK fits a quadratic function through multiple layers of an image pyramid using a FAST score, and finds maxima across the scale.

Blob detection is another of the popular ideas for feature detection. Convolution is in a certain way a localized comparison between two functions, convolving an image with a LoG filter (or its difference of Gaussian approximation) yields a blob detector [70]. Theoretically, performing this operation extracts the value of a smoothed version of the second derivative of the image, providing information about its brightness curvature. This information can also be obtained by analysing the eigenvalues of the Hessian matrix or its determinant [57], a fast approach to calculate these values using integral images was presented in [12]. Another blob detection method that takes advantage of integral images

is CenSurE (also known as the STAR detector), it works by calculating an approximation of the LoG at every pixel in the image [2].

Other approaches detect regions. The Maximally Stable Extremal Regions (MSER) works by thresholding the brightness and measuring the area of the connected binary regions [72]. When the area of a connected binary component does not vary much after a large variation in the threshold, it is considered an MSER region. Other methods for extracting regions are the super-pixels, these algorithms relay on energy optimization techniques [31, 100, 102, 1]. MSER regions, super-pixels and some symmetric-features are a few examples of non-convolutional feature detection. The blob detector introduced in chapter 4, is a recent instance of such non-convolutional techniques.

Tuytelaars and Mikolajczyk present a comprehensive survey on distinct methods for feature detection [99]. Their review includes a discussion on 'ideal' characteristics of features, it is recognised that repeatability is perhaps their most important trait. Features must be consistently detected despite several image transformations such as brightness, rotation, scaling and affinity.

With the intention of extracting information about affine transformations of the image, Mikolajczyk and Schmid introduced an iterative affine normalisation process over regions detected with the Harris or, the determinant of Hessian detectors [74]. This process generates a set of ellipses that contain information about affinity. MSER detector and the technique introduced in chapter 4 also generate ellipses around points of interest.

Alcantarilla *et al.* introduced a technique named KAZE [4]. The KAZE detector finds local-maxima in the determinant of Hessian response through non-lineal scale-spaces. An accelerated version of this approach was presented in [5].

Other methods make use of Machine Learning techniques to learn measures robust to image transformations. The Temporally Invariant Learned DEtector (TILDE), learns a regressor that is robust to strong changes in illumination [101]. Richardson and Olson optimize convolutional filters to perform stereo visual odometry [86].

2.1.2 Feature description

After the extraction of features, a more detailed representation of the image-patch surrounding such point is needed; perhaps the most widely used description algorithm is SIFT [65]. After calculating the dominant orientation and a scale, SIFT takes a squared area around the feature and measures the direction of the gradient within smaller squared regions known as bins; this approach is known as Histogram of Oriented Gradients (HOG). SURF [12] is another example of the HOG based descriptors.

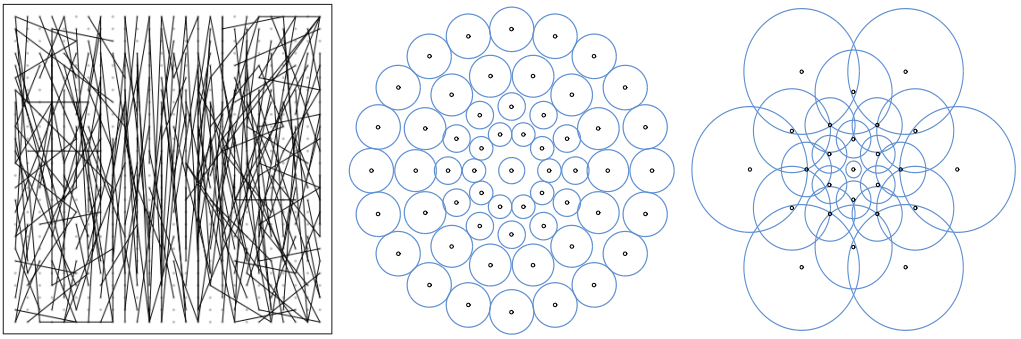
More recently a new set of efficient algorithms for description was introduced. Based on the comparison of intensities around a feature to produce a binary string, these approaches are denominated binary descriptors [39]. The main difference between the presented solutions is the pattern used to make the comparisons. ORB [88] finds uncorrelated patterns to improve on the randomly generated comparisons performed by BRIEF [18]. With a slightly more complex strategy, BRISK makes comparisons through different scales by smoothing the image with different Gaussian kernels [53]. In a similar fashion, FREAK compares intensities across scales by using a pattern inspired by the human retina [3]. Balntas *et al.* use an offline procedure to learn a robust binary descriptor optimised for each image-patch, this approach is named BOLD [9].

For a successful feature description, orientation must be normalised. Calculating the dominant orientation of a feature is a task usually performed in the detection stage using gradient estimation, the second-order moments or some other technique. Descriptors contain considerably more local information than the detector, BRISK and FREAK exploit this fact and calculate the orientation directly on the description stage.

The main advantage of binary over non-binary descriptors is that, a binary string can be compared very efficiently to another using the Hamming distance with an XOR operation and a bit count. Showing comparable performance results, binary descriptors can be much faster and consume much less memory resources than the standard approaches SIFT or SURF [39].

2.1.3 Mirror matching

The main objective of feature description, is to find corresponding regions across multiple images. Symmetry detection is not much different, mirrored versions of a region must be found across the same image. The naive approach to feature based symmetry detection is to detect, describe and match features across both, the original and a mirrored version of the image.



(a) ORB (b) BRISK (c) FREAK

Figure 2.3: Different binary descriptor sampling patterns.

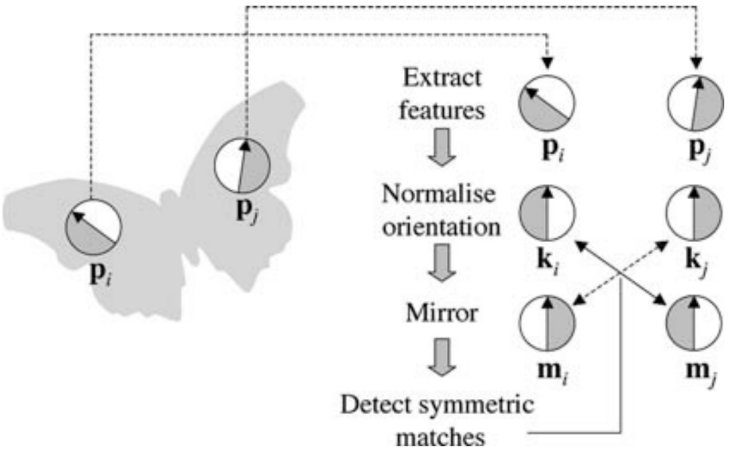


Figure 2.4: The process to find symmetric pairs of features in images. This image was taken from [66].

As suggested by Loy and Eklundh [66], it is possible to only find features and describe them in the original image to obtain the same results. Knowing the order in which the descriptor was constructed, one can modify the placement of the descriptor string to represent the same feature in a mirrored space.

Mirrored descriptors are compared against the non-mirrored versions to find local similarities. This method provides robustness against occlusions of symmetric regions. Figure 2.4 depicts the process of finding features, normalising orientation and mirroring to find symmetric pairs.

2.1.4 Parameter space

After finding corresponding regions in an image, it is necessary to extract information from such matches to find axes of symmetry. The Hough transform is a method that uses a voting process to find structures in images. For instance, the red line in figure 2.5 can be described by two parameters, its deviation angle ϕ form the horizontal axis, and its perpendicular distance r to the origin.

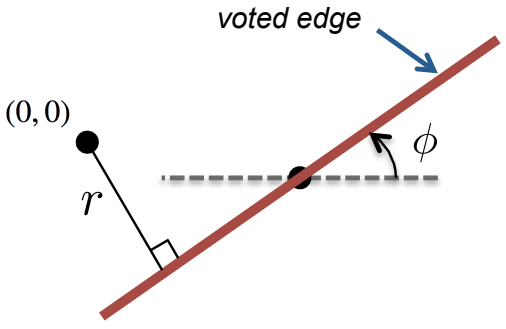


Figure 2.5: The Hough transform of a line.

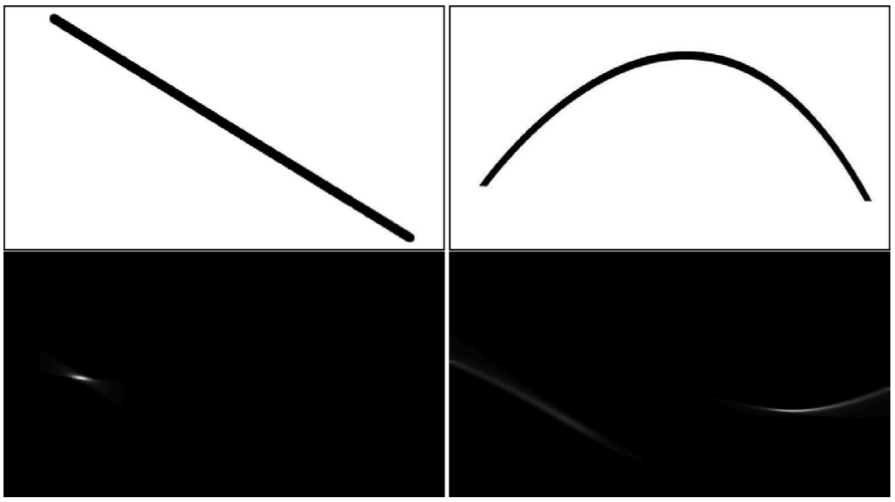


Figure 2.6: A straight line is a point in the parameter-space, whilst curved lines form contours.

In the Hough transform, ϕ is the gradient and r is the perpendicular distance to the centre of the image. Each pixel casts a vote in an accumulator matrix, the location of such vote is chosen from a discrete set of ranges depending on the values of the parameters. In the symmetric parameter-space two matched features vote for an axis of symmetry, such axis is perpendicular to the straight line that joins a pair of matched features. In chapter 3 a deeper explanation about the symmetry parameter-space is presented.

Figure 2.6 shows the accumulator matrix for both a straight and a curved line. The straight line can be recognised as a single point in the line parameter-space, while curved lines are smooth contours. In a similar fashion, a symmetric pair of matched features carries information about the axis of symmetry it supports.

In the original implementation, Loy and Eklundh weight the influence of every matched pair of features in the parameter-space. Penalising features of small scale and, distant and misaligned matches, their approach finds significant straight axes of reflection symmetry (figure 2.7). Lee and Liu [52] exploit the misalignment of matches and generate a third parameter to distinguish glide-reflection from pure reflection symmetry.

2.2 Conclusion

The conceptualization of symmetry as having different levels of complexity, explains the variety of approaches to solve its automated detection. Despite such abundance, symmetry detection is comparatively slow against other algorithms in computer vision, making it less attractive for implementation.

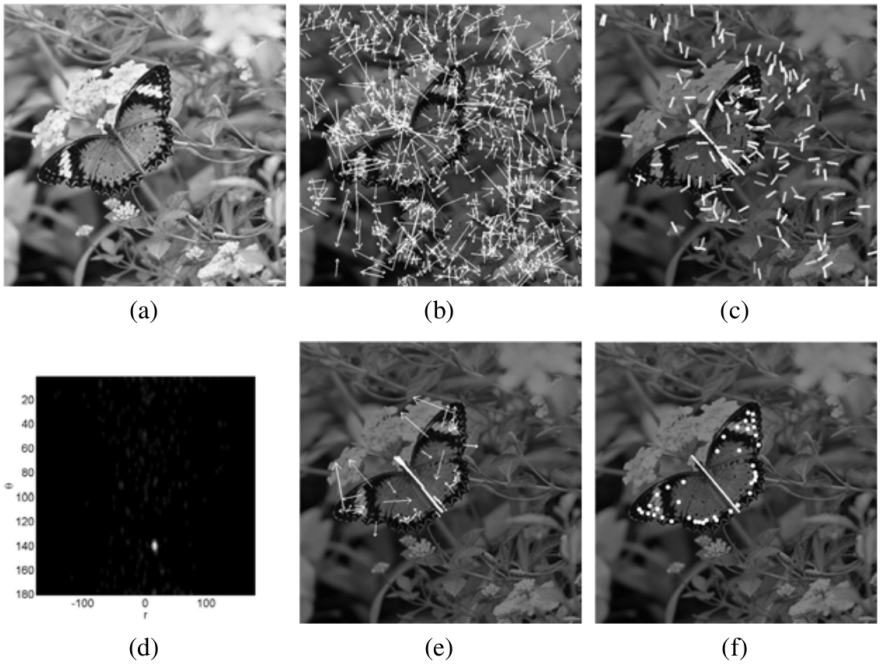


Figure 2.7: Loy and Eklundh's process for symmetry detection. This image was taken from [66].

Non feature based symmetry detectors are widely disrupted by textured structures. On the other hand, feature based detectors are inherently robust and perhaps assisted by such structures. These methods use multiple stages to find corresponding regions on an image. The first of them is the reduction of the search space by the means of feature detection. A drastic reduction of the search space may result in under-sampling (small numbers of detected features), and consequently cause false-positives or false-negatives. In contrast, large numbers of features benefit the detection of symmetry, however, this translates into slower computation times. For this reason, a tread-off between the number of detected features and the computation time must be found.

There are two main problems to tackle in feature based symmetry detection: the detection of large amounts of features and, the computational efficient ways to process them.

Chapter 3

A fast approach for feature-based symmetry detection

Feature based symmetry detection is computationally demanding. To achieve faster computation times, it is necessary to make every step of the process more efficient while maintaining the strengths that make symmetry detection operational and robust. Figure 3.1 recapitulates the steps for feature based symmetry detection. In this chapter the efficiency of feature description, mirroring and parameter-space analysis is addressed. A final regression step introduced in [52] is also shown in this chapter.

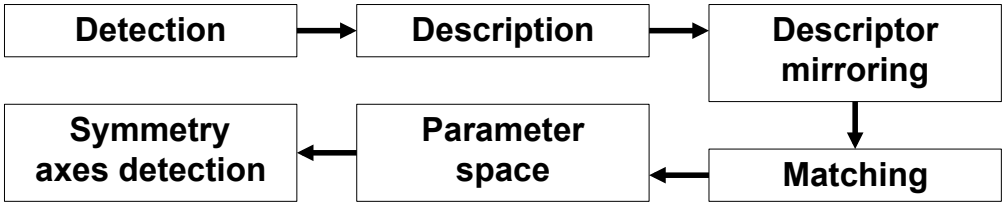


Figure 3.1: Flowchart for feature based symmetry detection.

3.1 Feature detection

The majority of the feature based schemes make use of a single feature detector. Lee and Liu use a feature detector in multiple transformed versions of the image [52]. They also discuss that abundance of detected features is a key factor for the successful detection of symmetry. To detect more features, Chertok and Keller also make use of multiple feature detectors [19]. For now, we will consider feature detection as a black-box that takes an image and outputs features $\mathbf{p}_i = (x_i, y_i, \phi_i, s_i)$, where (x_i, y_i) represents the location, ϕ_i the rotation and, s_i the scale of the i^{th} feature.

3.2 Mirror binary descriptors

The usual strategy for the description stage amongst symmetry detectors is SIFT [66, 79]. Binary descriptors are considerably faster to calculate while providing similar accuracy

results [39]. We use a BRISK [53] descriptor with 57 samples and 512 comparisons $\mathbf{b} = \{b_p \mid p = 0, 1, \ldots, 511\}$, each comparison between two samples around a feature can be represented with a single bit value.

The traditional BRISK pattern is slightly modified, we make the pattern symmetric across the orientation of the feature and index it in such a way that b_p is the mirror pairing of b_{511-p} across the feature orientation. Figure 3.2 shows the modified pattern, the black arrow denotes the feature orientation.

The 512 comparisons in the symmetric descriptor are stored as an array of bytes with $512/8 = 64$ elements. To mirror the descriptors, each byte is efficiently bit-swapped using a look-up table; for instance, the mirror version of the byte 0xB2 = 0b10110010 is 0x4D = 0b01001101. Then, the order of the array is reversed; this process is shown in figure 3.3.

The Hamming distance between two binary descriptors $\Delta(\mathbf{b}_i, \mathbf{b}_j)$, can be calculated with an XOR and a bit-count. A matching pair $\mathbf{p}_{ij} = (\mathbf{p}_i, \mathbf{p}_j)$ exists if the Hamming distance between the descriptors of $\mathbf{p}_i$ and $\mathbf{p}_j$ is less than a threshold *i.e.* ,

$$\exists\, \mathbf{p}_{ij} \iff \Delta(\mathbf{b}_i, \mathbf{b}_j) < BRISK_{thresh}. \tag{3.1}$$

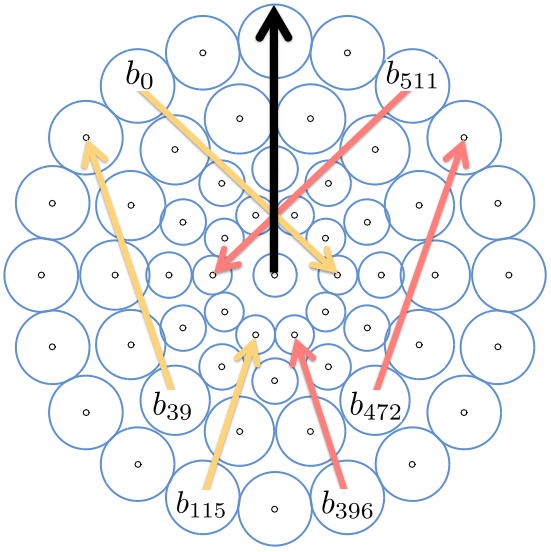


Figure 3.2: Comparisons made across the BRISK pattern are indexed in a symmetric manner along the orientation of the feature (denoted with a black arrow).

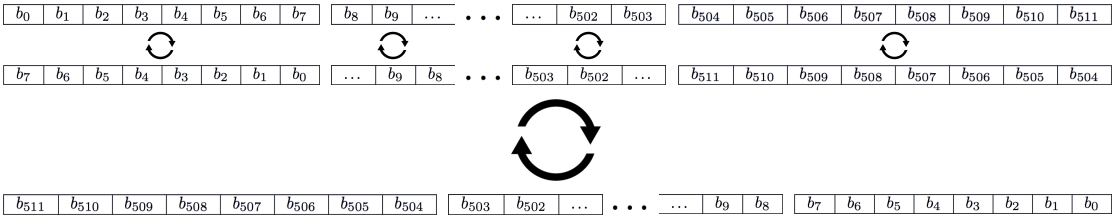


Figure 3.3: The symmetric arrangement of the BRISK descriptor allows for a very efficient mirroring. This is achieved by bit-swapping bytes with a look-up table and reversing the order of the array.

3.3 Parameter space

A parameter-space is created from the information provided by successful matches *i.e.* point pairs $\mathbf{p}_i$ and $\mathbf{p}_j$ whose Hamming distance is less than $BRISK_{thresh}$. We use a two-dimensional parameter-space with coordinates (ϕ_{axis}, r_{ij}) calculated with equations 3.2, 3.3 and 3.4 (figure 3.4). In this 2D parameter-space, pure reflection symmetry axes are single points whilst curved reflection axes are smooth contours (figure 3.5). For the construction of the parameter-space, only the location and the orientation of the features is accounted for, the scale is disregarded *i.e.* $\mathbf{p}_i = (x_i, y_i, \phi_i)$.

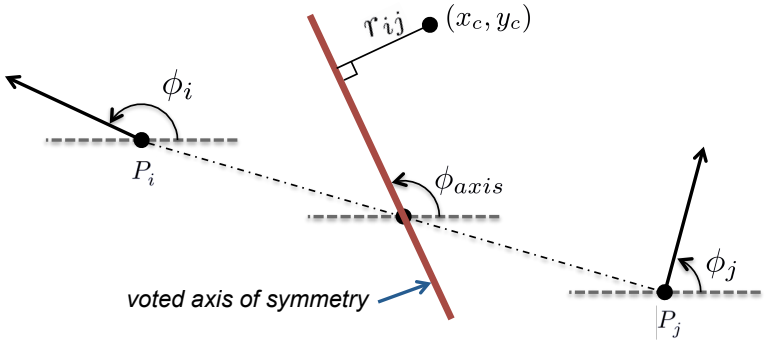


Figure 3.4: The symmetry parameter-space. Every matched pair of features $\mathbf{p}_{ij} = (\mathbf{p}_i, \mathbf{p}_j)$, is mapped to a point in the parameter-space with coordinates (ϕ_{axis}, r_{ij}).

The angle in equation 3.2 is defined in the range $\{-\pi \leq \phi'_{axis} < \pi\}$. As we only care for the angle of the axis and not the direction in which it points, we use equation 3.3 to re-bound this parameter in $\{0 \leq \phi_{axis} < \pi\}$. The second coordinate of the parameter-space is the perpendicular distance between the axis and the centre of the image with coordinates (x_c, y_c).

$$\phi'_{axis} = \frac{\phi_i + \phi_j}{2}. \tag{3.2}$$

$$\phi_{axis} = \begin{cases} \phi'_{axis} & \text{if } \; 0 \leq \phi'_{axis} \\ \phi'_{axis} + \pi & \text{if } \; \phi'_{axis} < 0 \end{cases}. \tag{3.3}$$

$$r_{ij} = \left(\frac{x_i + x_j}{2} - x_c\right) sin(\phi'_{axis}) - \left(\frac{y_i + y_j}{2} - y_c\right) cos(\phi'_{axis}). \tag{3.4}$$

3.3.1 Density-based symmetry axes detection

In other approaches, each sample casts a vote in an accumulator matrix. This accumulator matrix is a discrete approximation of a density map of the parameter-space, where points of local-maxima represent axes of symmetry. To be able to detect such points, the accumulator matrix has to be smoothed and then a non-maxima suppression algorithm is used. The set of successful matches that generated a point of high-density represents an axis of symmetry, this procedure is appropriate for straight axes as these are single points in the parameter-space. Curved axes of symmetry are contours in the parameter-space; therefore the single-point detection approach is no longer suitable (figure 3.5). We consider now an axis of symmetry to be a dense cloud of points in the parameter-space.

We use a density-based clustering algorithm called DBSCAN to detect dense clouds of points [28]. Consider the following classification:

- Core point: if there are at least $MinPts$ points within an ϵ distance of a point, such point is considered as a core.
- Directly reachable point: a point is directly reachable from a core point if they are within an ϵ distance from each other.
- Density reachable point: two points are density reachable if there is a path of directly reachable core points that connects them.
- Outliers: all points that are not directly reachable from a core point and are not core points themselves, are considered as outliers.

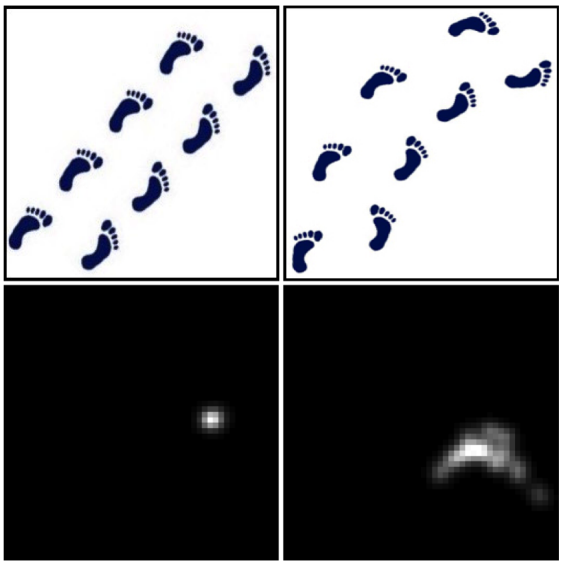


Figure 3.5: The difference between straight and curved axes of symmetry in the parameter-space. Straight axes are single points in the parameter-space, whilst curved axes are contours.

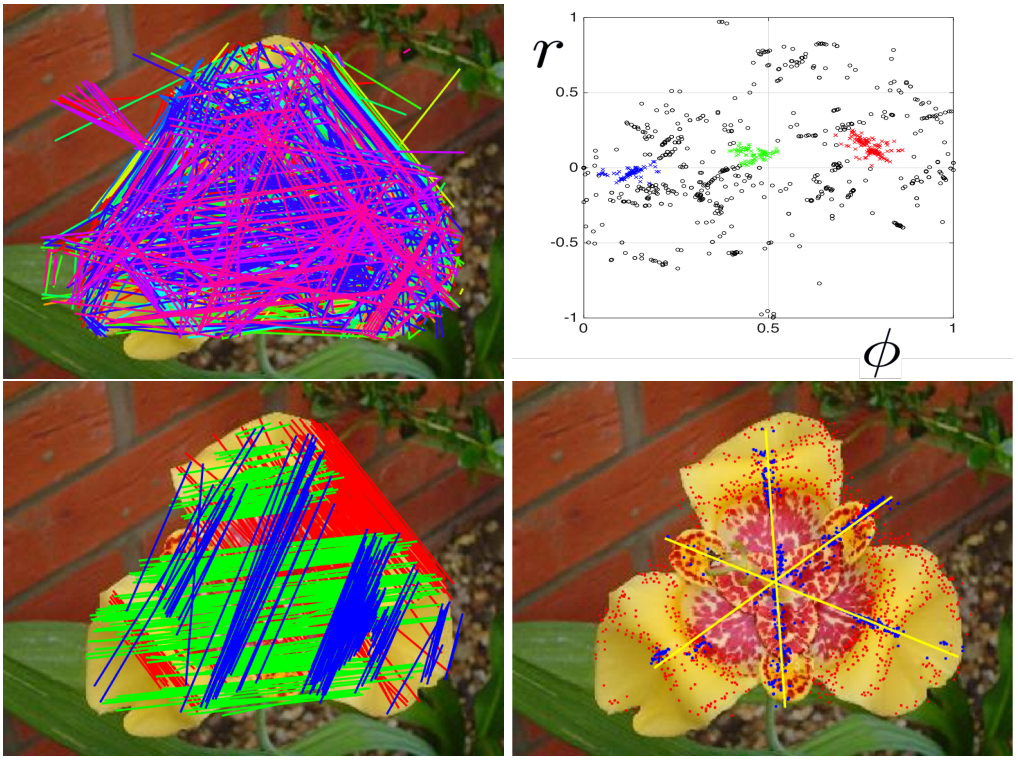


Figure 3.6: Multiple axes detection example. On the top-left all successful matching pairs are shown. Three regions of high density of points are detected in the normalised parameter-space that represent axes of symmetry, blue, green and red (top-right and bottom-left). Outliers are discarded by DBSCAN, these are marked with black in top-right. Optimal axes are extracted from the clustered points and shown in the bottom-right.

DBSCAN works by visiting a sample and expand through other density-reachable points. A cluster is complete when there are no more density reachable points from its boundaries, all samples that have been labelled to belong to a cluster are not revisited; this reduces the complexity of the algorithm from $O(n^2)$ to $O(nlog(n))$ (in the best case scenario). DBSCAN takes two parameters, ϵ indicates how densely clustered are the points and, $MinPts$ indicates the minimum number points for a cluster to exist.

Figure 3.6 shows clustering results for the multiple axes detection task, matched features before and after clustering are also shown. The parameter-space is normalised in the range $\{0 \leq \phi_{axis} \leq 1, -1 \leq r_{ij} \leq 1\}$ as the clustering relies on the Euclidean distance of the points.

This algorithm does not need to know the number of clusters to be found and it accounts for outliers. These two are the most important traits that aid the proper detection of symmetry. The number of symmetric objects in an image is usually unknown, DBSCAN allows the detection of multiple axes without added computation time. For the single-axis detection task, only the cluster with the largest number of samples is considered. The rejection of outliers de-noises the clusters, improving the quality of the axes.

DBSCAN divides the set of all successful matches $\mathbf{p}_{ij}$ (denoted as $\mathbf{P}$), into L clusters $\mathbf{R}_l$ for $l = 1, \ldots, L$, that represent the axes and, an extra subset $\mathbf{O}$ representing the outliers. The next condition is met,

$$\mathbf{O} \cup \mathbf{R}_1 \cup \mathbf{R}_2 \cup \cdots \cup \mathbf{R}_L = \mathbf{P}. \tag{3.5}$$

3.4 Optimal axes

In this section the elements of $\mathbf{R}_l$ need to be indexed, now denoted as $\mathbf{p}_{ij,a}$ for $a = 1, 2, \ldots, A$; where A is the cardinality of $\mathbf{R}_l$.

Optimal axes of symmetry are found with a polynomial regression on the middle points of all $\mathbf{p}_{ij,a} \in \mathbf{R}_l$, calculated with equation 3.6; this process was introduced in [52].

$$\begin{bmatrix} \mathbf{p}_{ij,a,x} \\ \mathbf{p}_{ij,a,y} \end{bmatrix} = \begin{bmatrix} \frac{x_i + x_j}{2} \\ \frac{y_i + y_j}{2} \end{bmatrix}. \tag{3.6}$$

The polynomial regression of degree d is shown in equation 3.7.

$$\begin{bmatrix} \mathbf{p}_{ij,1,y} \\ \mathbf{p}_{ij,2,y} \\ \vdots \\ \mathbf{p}_{ij,A,y} \end{bmatrix} = \begin{bmatrix} 1 & \mathbf{p}_{ij,1,x} & \mathbf{p}_{ij,1,x}^2 & \cdots & \mathbf{p}_{ij,1,x}^d \\ 1 & \mathbf{p}_{ij,2,x} & \mathbf{p}_{ij,2,x}^2 & \cdots & \mathbf{p}_{ij,2,x}^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \mathbf{p}_{ij,A,x} & \mathbf{p}_{ij,A,x}^2 & \cdots & \mathbf{p}_{ij,A,x}^d \end{bmatrix} \begin{bmatrix} q_0 \\ q_1 \\ \vdots \\ q_d \end{bmatrix} + \begin{bmatrix} z_0 \\ z_1 \\ \vdots \\ z_A \end{bmatrix}. \tag{3.7}$$

In matrix notation,

$$\mathbf{y} = \mathbf{X}\mathbf{q} + \mathbf{z}. \tag{3.8}$$

The vector $\mathbf{q}$ contains the coefficients of the polynomial of degree d that best describes the data. $\mathbf{z}$ is the error between the samples and the regressed model. In the sense of the least squares, equation 3.8 is solved for $\mathbf{q}$ using equation 3.9. Equation 3.9 holds true only if $\mathbf{X}$ has at least $d + 1$ linearly independent rows.

$$\mathbf{q} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{3.9}$$

To calculate the regression in different angles σ we rotate the middle points using equation 3.10. Each polynomial regression is performed for $d = 1, 2, 3, 4, 5$ and $\sigma = 0°, 45°, 90°, 135°$.

$$\begin{bmatrix} \mathbf{p}_{ij,a,x,\sigma} \\ \mathbf{p}_{ij,a,y,\sigma} \end{bmatrix} = \begin{bmatrix} cos(\sigma) & -sin(\sigma) \\ sin(\sigma) & cos(\sigma) \end{bmatrix} \begin{bmatrix} \mathbf{p}_{ij,a,x} \\ \mathbf{p}_{ij,a,y} \end{bmatrix}. \tag{3.10}$$

Out of the twenty axes regressions, we consider as optimal the one with the smallest root-mean-square error *i.e.* the smallest value of $||\mathbf{z}||$. Figure 3.6 shows the optimal axes from the clusters detected in an image of a flower.

3.5 Conclusion

State of the art symmetry detectors use description methods like SIFT [79], contour curvature histograms [6], or affine-invariant contour features [104], to name a few. Such operations involve time consuming memory access and processing of long floating-point vectors. For instance, SIFT uses 128 floating-point values for a total of $128 * 3 = 384$ bytes; on the other hand our symmetric BRISK descriptor has only $512/8 = 64$ bytes, and uses bitwise logical operators. This provides a significant reduction of the computation time (discussed in chapter 6). Calculation of the Hamming distance and memory access times can be further improved depending on the architecture being used *e.g.* 32 or 64 bits.

Moreover, the detection of axes in the parameter space with a density-based approach provides three main advantages with a single solution: multiple axes detection, curved and straight axes detection and, it accounts for outliers.

The forthcoming chapters make two proposals for the feature detection stage. The first addresses the rapid detection of features to reduce computation times, and the second tackles the detection of multiple classes of features with a single method.

Chapter 4

Locally contrasting keypoints

In this chapter a new non-deterministic algorithm for affine-covariant blob detection is introduced, we show how it compares to other modern feature detectors. A variant to the approach is also presented that better detects larger structures. The main advantages of the new approach over other detectors are, computation speed and the dispersion of features across the image. Figure 4.1 shows the detected features on a real-world image.

This novel method has two main components. The image is first transformed into a feature map using a non-deterministic voting process named the *Brightness Clustering Transform* (BCT). The BCT benefits from the use of integral images to perform a fast search through different scale spaces. Information from the maps is then extracted, the detected features are called the *Locally Contrasting Keypoints* (LOCKY). LOCKY features capture information about the scale and affine transformations, and are up to three times faster to detect than the MSER regions.

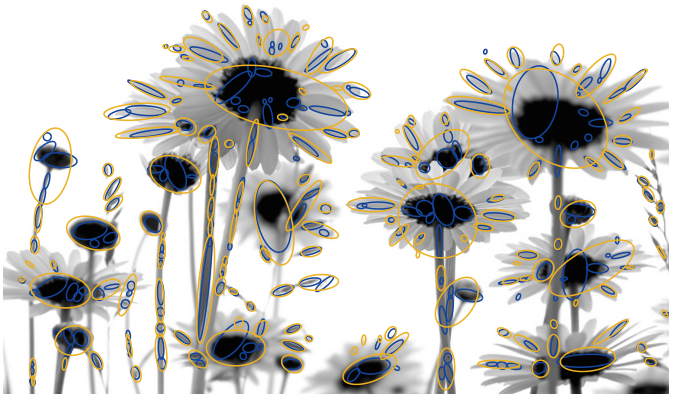


Figure 4.1: LOCKY (blue) and LOCKY-S (yellow) blobs detected on an image of daisies.

Feature detectors are called *invariant* to some transformations, Tuytelaars and Mikolajczyk suggest the term should be *covariant* when they change covariantly with the image transformation [99]. Mikolajczyk *et al.* [74] present an affine-covariant solution that, based on the second order moments of the regions surrounding the detected points, the features are affine-normalised; their solution is robust and elegant but very slow. MSER is also affine-covariant, it improves the computation time over Mikolajczyk's work, although it lacks robustness against blurring and noise. The use of affine features is related to the intention of making detectors robust to perspective changes. Human vision is extremely

resilient to image transformations like blurring, rotation and scaling, and also to changes in perspective. Although perspective transformations are different from affine transformations, affine-covariant feature detectors are good approximations due to their local nature.

Integral image Our approach uses integral images to speed up the process of blob detection. The integral image as presented by Viola and Jones, is a very useful tool to calculate the sum of rectangular areas of pixels in only three operations disregarding the size of the rectangles; it is widely used because it allows to make calculations at different scales without added computational cost [103]. Equation 4.1, shows the definition of the integral of the image function $f(x, y)$. The recurrences in equations 4.2 and 4.3 allow the calculation of $ss(x, y)$ in one pass over the image ($s(x, y)$ is the cumulative row sum, $s(x, -1) = 0$ and $ss(-1, y) = 0$).

$$ss(x, y) = \sum_{x' \leq x, y' \leq y} f(x', y'). \tag{4.1}$$

$$s(x, y) = s(x, y - 1) + f(x, y). \tag{4.2}$$

$$ss(x, y) = ss(x - 1, y) + s(x, y). \tag{4.3}$$

4.1 The brightness clustering transform

Most of the complex human visual activities require alternations between eye fixations and significantly rapid eye movements known as saccades. The information humans extract from a scene is a series of snapshots (fixations), however, we perceive scenes as single views [43]. There exist a temporal memory which integrates those snapshots into one *memory image*, preserving the extracted information from the fixations; such process is known as *trans-saccadic integration.*

The BCT is a non-deterministic algorithm that employs a voting scheme using rectangles on the integral image space. Each vote is randomly initialised, then it extracts information from the region being attended resembling eye fixations. When the next vote is initialised, the algorithm changes the location of attention, this is a saccade.

4.1.1 Blob detection

The BCT begins with an accumulator matrix of the same size as the input image (the output *memory image*). A vote means incrementing the value of an element in the accumulator matrix by one; each vote is obtained in three steps. First, a rectangle with

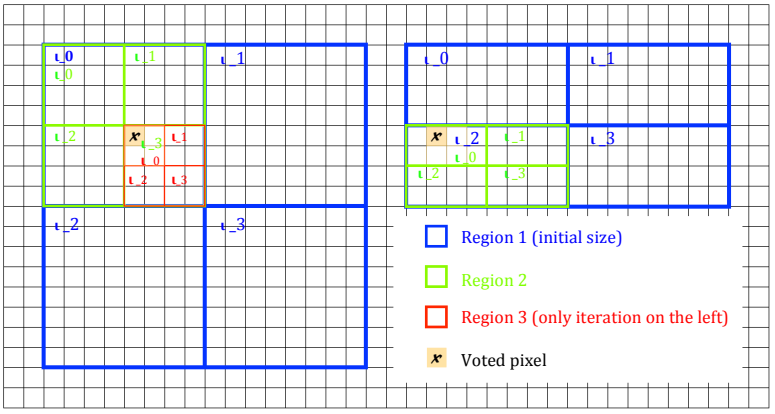


Figure 4.2: A squared vote on the left and a rectangular vote on the right. I_0 in blue, I_1 in green and I_2 in red. The subregions of every step are marked as ι_i with the same colour as the step they belong to.

random position and size is initialised within the image. For finding blobs, we select the width and height of the rectangle to be a power of two *i.e.* $width = 2^u$ and $height = 2^v$ $\{u, v \in \mathbb{N}\}$. The second step is to divide the rectangle in four smaller sub-regions, the sub-region with the biggest sum of intensities is now considered to be the initial rectangle; the sum is calculated using the integral image. Consider the rectangle I_t where $t = 0$ for the initial position and size, and its subregions ι_0, ι_1, ι_2 and ι_3; the next region will have an initial rectangle I_{t+1}. The second step is repeated until I_t has either $width = 2$ or $height = 2$.

$$I_{t+1} = \arg\max_{\iota_i} \sum_{x,y \in \iota_i} f(x, y), \quad i = 0, 1, 2, 3. \tag{4.4}$$

Suppose the last I_t is situated in (x_f, y_f), has $width = w_f$ and $height = h_f$; in the third step, the pixel in $loc = (round(x_f + w_f/2), round(y_f + h_f/2))$ is voted. This sequence of steps is graphically presented in figure 4.2, algorithm 1 shows the pseudocode for the BCT. After a user-defined number of votes (denoted with V), the accumulator matrix is smoothed with a small Gaussian kernel with $\sigma = 2$ and then normalised. Smoothing removes the effects of noise in the voting process and helps to find the true shape of the extracted blobs.

The BCT can be thought as a coarse-to-fine search through scale spaces for local maxima of the image. Consider the first-derivative Haar-like wavelets in figure 4.3. Within the rectangle in question, choose the white side if the response to such filter is positive, or the black side if it is negative. The intersection of the two selected areas (one for each wavelet) yields the highest sum of pixel brightness; this is also the direction of the image gradient. Thus, by selecting the subregion ι_i with the highest sum of intensities, we are selecting the subregion that gradient points to. Every sub-division of a rectangle is in the next smaller octave and thus, the votes start at a big scale and refine until they get to the smallest scale possible on the image.

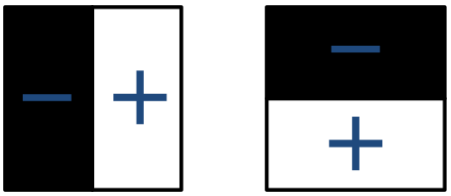


Figure 4.3: First derivative Haar-like wavelets, useful to rapidly approximate the gradient of the image. The ι_i with the highest sum of intensities is the subregion that gradient points to.

The use of rectangles benefits affine locality; for example, a horizontal rectangle discards some information in the y axis and operates in the same scale in the x axis, this allows an improvement on the detection of oval blobs. Suppose a vote lies in (x_f, y_f) with an initial rectangle in (x_0, y_0), another vote will most likely lie in $(x_f + 1, y_f)$ if a rectangle of the same size is set in $(x_0 + 1, y_0)$. This property clusters votes around the centre of blobs, consequently the shape of the blobs is extracted.

Amounts ranging from 5×10^4 to 1×10^5 votes suffice to extract blobs in a 1024×768 image. We also noted that amounts as small as 2×10^4 votes can extract most of the

Algorithm 1 BCT.

$integ$ = calculate integral of the input image
$accum$ = initialise accumulator matrix

for $(vote = 1$ **to** $V)$ **do**

First Step:
Init region I:
$u = \text{rand}(rangeMin, rangeMax)$
$v = \text{rand}(rangeMin, rangeMax)$
$width = 2^u$
$height = 2^v$
$x = \text{rand}(0, imWidth - width)$
$y = \text{rand}(0, imHeight - height)$

Second Step:
while $(width > 2 \ \& \ height > 2)$ **do**

divide I in 4 subregions $\iota_0, \iota_1, \iota_2$ and ι_3
ι_{max} = max-brightness$(\iota_0, \iota_1, \iota_2, \iota_3)$ (use $integ$)
$I = \iota_{max}$ this implies:
$width = width/2$
$height = height/2$
$x = x_{\iota_{max}}$
$y = y_{\iota_{max}}$
end while

Third Step:
$loc = (x + width/2, y + height/2)$
$accum[loc] = accum[loc] + 1$

end for

Algorithm 2 BCT-S.

Modified Second Step:

for $(div = 0;\ div < 3;\ div++)$ **do**

divide I in 4 subregions $\iota_0, \iota_1, \iota_2$ and ι_3
ι_{max} = max-brightness$(\iota_0, \iota_1, \iota_2, \iota_3)$ (use $integ$)
$I = \iota_{max}$ this implies:
$width = width/2$
$height = height/2$
$x = x_{\iota_{max}}$
$y = y_{\iota_{max}}$

end for

Modified Third Step:

$accum[\iota_{max}] = accum[\iota_{max}] + 1$

significant blobs on the same image. The most commonly used values for the width and height of the rectangles range from 2^3 to 2^7; this range may be modified depending on the size of the image and the size of the blobs to be extracted.

So far the bright blobs are extracted by finding the sub-regions with the biggest sum of pixels; if we want to find dark blobs we could either modify equation 4.4 to be

$$I_{t+1} = \arg\min_{\iota_i} \sum_{x,y \in \iota_i} f(x,y), \quad i = 0,1,2,3, \tag{4.5}$$

or, find the bright blobs of the inverted image *i.e.* considering the image is an 8-bits representation, we do $f'(x,y) = 255 - f(x,y)$.

Continuing the voting process while the width and the height are greater than two, concentrates the votes in a compact manner around locally contrasting regions. This compactness achieves good shape description of small regions, therefore the detection of larger features in images with higher resolution is difficult.

Sometimes fine features are desired, when this is not the case the second and third steps can be modified to be able to detect larger features. An extension of the BCT for extracting larger structures (BCT-S) is described in algorithm 2. In the second step, instead of iterating the rectangle reduction until either the width or the hight equal two, the reduction is only performed three times ($rangeMin > 2$). On the third step, all the pixels within the last ι_{max} region are voted. This is the equivalent to running the algorithm at multiple scales. Figure 4.4 shows the detected features using the BCT and the BCT-S.

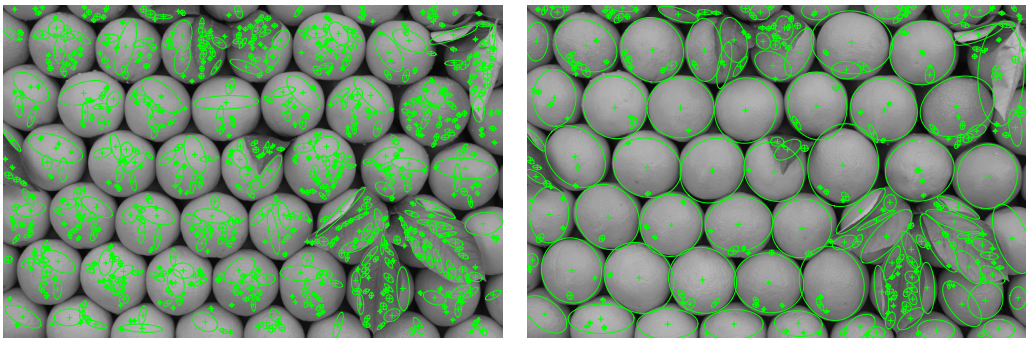


Figure 4.4: LOCKY features on the left (extracted using the regular BCT), and LOCKY-S features on the right (extracted using the BCT-S). The BCT-S promotes the detection of larger structures in the image.

4.1.2 Ridge detection

Ridges are one-dimensional blobs, the BCT can be modified to extract these features. Half of the votes are obtained with the width set constant and equal to two, and the height again set to a power of two; this time, the rectangles are only divided in two parts along the height. The other half of the votes are obtained in the same manner, but setting the height to be constant and equal to two and varying only the width.

Figure 4.5 shows the difference between blob and ridge detection using the BCT. This operator yields a new fast ridge detector that can be used for applications like matching, extraction or segmentation.

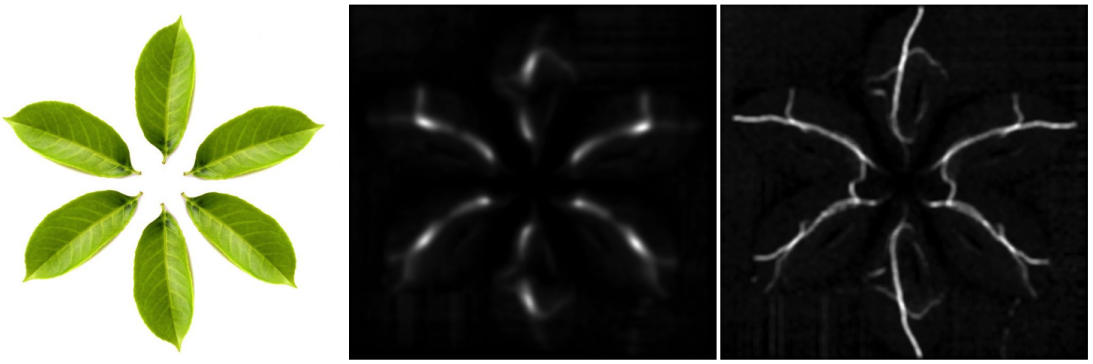


Figure 4.5: The BCT can be modified to create ridge maps (right). Blobs are small regions (middle), while ridges are contour-like structures.

4.1.3 Operator invariance

The scale invariance of the BCT, is related to the size of the initial rectangles; as the voting process goes on, the centres of the blobs are found. The random aspect ratio of the rectangles helps to disperse the votes near the centre of the features and not only cluster them at the exact centre.

Changes in illumination are considered as multiplicative noise. We modify equation 4.4 as follows.

$$I_{t+1} = \arg\max_{\iota_i} \sum_{x,y \in \iota_i} g(x,y) f(x,y). \tag{4.6}$$

Here, $g(x,y)$ is the illumination function. If the illumination function can be considered to be approximately constant *i.e.* $g(x,y) \approx g$ in the region covered by I_0, then g can be extracted from the equation and we see that the operator is invariant to approximately constant changes in illumination.

From the same logic we obtain that the BCT is invariant to noise with zero-mean distributions as it cancels itself in the sub-regions ι_i.

4.2 Locally contrasting keypoints

The Locally Contrasting Keypoints (LOCKY) are blob features extracted directly from the accumulator matrix of the BCT. After the normalisation process, the accumulator matrix is thresholded; the set of all the connected components in the binary image, are the detected blobs. Figure 4.6 describes the process for detecting the LOCKY blobs in an image.

Fitting an ellipse with the same second order moments as the connected components is a fast way of extracting information from them. With a $2 \times M$ matrix ($M > 1$) containing the coordinates (x_m, y_m) of the pixels in a connected component; the mean is the centre of the feature (equation 4.8).

$$\mathbf{Q} = \frac{1}{M-1} \sum_{m=1}^{M} ((x_m, y_m) - (\mu_x, \mu_y))((x_m, y_m) - (\mu_x, \mu_y))^T. \tag{4.7}$$

$$(\mu_x, \mu_y) = \frac{1}{M-1} \sum_{m=1}^{M} (x_m, y_m). \tag{4.8}$$

The eigenvalues of the sample covariance matrix $\mathbf{Q}$ (equation 4.7) represent the size of the axes of the ellipse with the same second order moments; the eigenvectors define the direction of the axes. In practice, the eigenvalues of $\mathbf{Q}$ are scaled up by a factor of five to enlarge the size of the features.

This step is similar to the ellipses of the MSER Regions. The advantages of this method are that one can detect the scale of the features by the size of the axes and, it is also possible to extract information about affine transformations and rotation of the blobs.

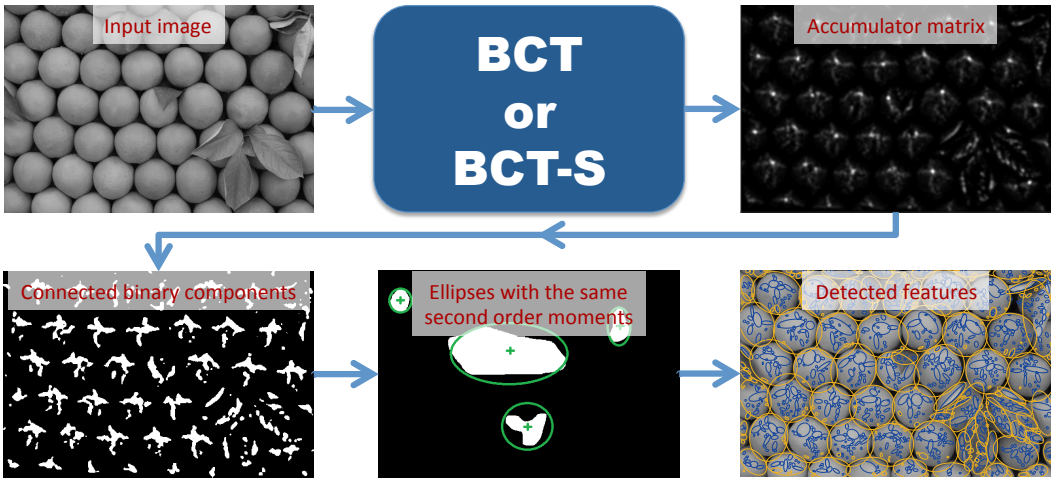


Figure 4.6: The process for extracting the LOCKY features begins with the transformation of the image into a blob map using the BCT (blue) or the BCT-S (yellow). The accumulator matrix is thresholded and the set of connected components is then extracted. The LOCKY features are the ellipses with the same second order moments as the connected binary components.

4.2.1 Time complexity analysis

The time complexity of the algorithm is mainly dependent on two stages:

- The BCT or the BCTS. The BCT requires the computation of the integral image, this is achieved in $2X$ operations where X is the number of pixels in the image. Each iteration of a vote involves nine value accesses on the integral image, four sums, eight subtractions and, four comparisons. The BCT has linear complexity on V. The regular BCT reduces the rectangles to a minimum size and aggregates only one pixel in the accumulator matrix. On the other hand, the BCTS reduces the rectangles three times and aggregates a number of pixels depending on $rangeMin$ and $rangeMax$. After the voting process, the accumulator matrix is smoothed with a small Gaussian kernel; convolution is of linear complexity on X. The time complexity of this stage is then $O(X)$.
- The extraction of features from the accumulator matrix. Dealing with one connected component at a time, its labelling and the ellipse fitting step can be computed in linear time on X.

The complexity of LOCKY is therefore $O(X)$. Other algorithms perform an extensive search through scale-spaces at every pixel using convolutions or other methods. MSER on the other hand, does not use convolution, its operation relies on the detection of connected binary components at several thresholds of the image. On the original implementation, this is achieved by using a union-find data structure with path compression;

this approach is known to have $O(X\alpha(X))$ complexity, where $\alpha(X)$ is the inverse Ackermann function. Nistér and Stewénius present an implementation of MSER that runs in linear time by working with only one connected component at a time [78].

4.3 Detection results

The Oxford affine-covariant regions dataset [75], presents a good challenge for feature detection and it is widely used for evaluation; eight sequences composed of six images each with increasing image transformations including decreasing illumination, change in perspective, blurring, jpeg compression and a mix of scale and rotation (figure 4.7). We use the measure of *repeatability* defined in the same paper to compare LOCKY and LOCKY-S with both, affine-covariant and non affine-covariant detectors.

In the affine-covariant group we use as comparison the *Harris-Affine* and *Hessian-Affine* detectors [74] and *MSER* [72]. In the non affine-covariant group, we use the BRISK detector [53], the SURF (fast-Hessian) detector [13] and, the STAR detector [2]. LOCKY-1 uses 1×10^5 rectangles of size ranging from 2^3 to 2^5 and a threshold of 12%; LOCKY-2 uses 1×10^6 rectangles of the same characteristics. Both LOCKY-1 and LOCKY-2 use the regular BCT; LOCKY-S uses the same settings as LOCKY-1 with the only difference that it uses the BCT-S to extract the features.

The measure of repeatability consists of projecting the detected features to the same basis as the original image (the first image of the sequence) using an homography matrix; comparing correspondences of features and measuring how well the region detected on the images overlap. For more information on this measure see [75]. We use the correspondences with 40% overlap. To be able to compare LOCKY and LOCKY-S with non-affine-covariant detectors we "disable" the measure by using a circle with a radius equal to half the size of the major axis of the ellipses, these results are shown in figure 4.9. Since the detection of the LOCKY features is based in a non-deterministic transform,

Figure 4.7: The first images of the sequences in the Oxford affine-covariant regions dataset [75].

Detector	Time (ms)	Type	Affine	Disp.
SURF [13]	95.1	Blobs	✕	12.95
BRISK [53]	20.5	Corners	✕	10.42
STAR [2]	15.1	Blobs	✕	12.57
LOCKY-1	21.0	Blobs	✓	1.26
LOCKY-2	100.6	Blobs	✓	1.65
LOCKY-S	22.0	Blobs	✓	1.04
MSER [72]	68.5	Regions	✓	8.67
Hessian-Affine [74]	126.8	Blobs	✓	19.03
Harris-Affine [74]	179.1	Corners	✓	16.16

Table 4.1: The average computation time for a set of 127 images, expressed in milliseconds. The images were converted to grayscale with 1024×768 pixels. The dispersion index shows how scattered the features are across the image (calculated with equation 4.11 averaged over the same set of 127 images, the lower the index the more evenly-distributed).

every run matches different features; the repeatability results of LOCKY-1, LOCKY-2 and LOCKY-S include the mean and variance over 100 runs of the test (examples of the detected features are shown in figure 4.8).

Repeatability results for LOCKY demonstrate its robustness against blurring and changes in brightness *i.e.* the repeatability score is kept approximately constant throughout these image transformations. Moreover, LOCKY seems to perform better in images with smaller rectilinear structures such as the wall sequence. The decay of repeatability across image transformations seems similar, sometimes better than other detectors.

The timing results shown in table 4.1, were obtained using the OpenCV implementations of the algorithms (using mex files in MATLAB) with a 2 GHz *i*7 processor[1][2]. Note that BRISK uses a multi-scale version of the AGAST detector which is a faster implementation of the FAST detector; LOCKY and LOCKY-S have similar timings while being able to provide information on affine transformations of the features.

In 2008, Tuytelaars and Mikolajczyk suggested that the density of features should reflect the information content of the image [99]. This is a useful property for tasks such as object detection; nevertheless, for tasks such as navigation or landmark matching, occlusion can become a big problem if features cluster on regions of the image [118]. Spacial dispersion of features and the use of other evaluation measures were addressed in [39]. A measure of statistical dispersion is a non-negative real value that is zero when all samples are equal, and increases as the data becomes more varied *e.g.* variance, coefficient of variation or entropy.

The value D presented in table 4.1, is obtained by dividing the image into $B = 100$ non-overlapping bins. We count the number of features that lie in each bin as a two-dimensional histogram of locations. If the location of a feature is represented with a two

[1] Harris-Affine and Hessian-Affine from http://www.robots.ox.ac.uk/∼vgg/research/affine
[2] Code for LOCKY is made publicly available in https://github.com/jimmylomro/locky

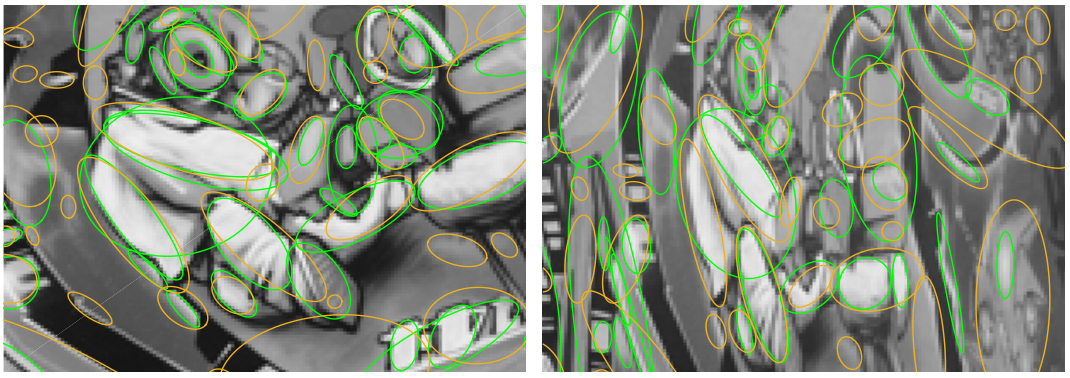


Figure 4.8: LOCKY-S features in yellow and MSER regions in green, on the second (left) and fifth (right) images of the graffiti sequence. The detected features change covariantly with the change in perspective.

dimensional vector $\mathbf{p}_i = (x_i, y_i)$ having P features, the count of features in each bin (a region called Γ_b) is represented by C_b (equation 4.9).

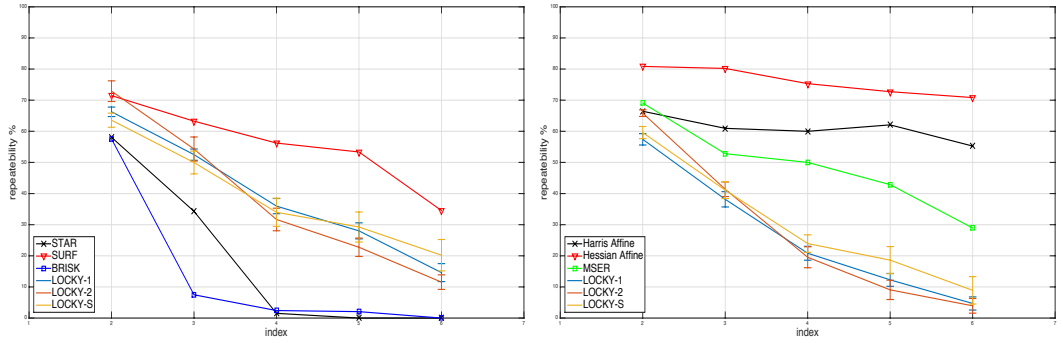
$$C_b = \sum_{i=1}^{P} a_b(\mathbf{p}_i), \tag{4.9}$$

$$a_b(\mathbf{p}_i) = \begin{cases} 1 & \text{when } \mathbf{p}_i \subset \Gamma_b, \\ 0 & \text{otherwise.} \end{cases} \tag{4.10}$$

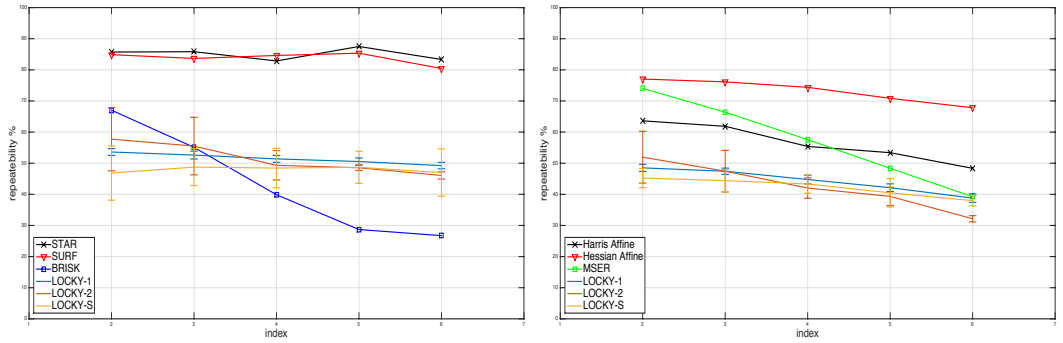
The index D is calculated as

$$D = \frac{\sum_{b=1}^{B}(BC_b - N)^2}{N}. \tag{4.11}$$

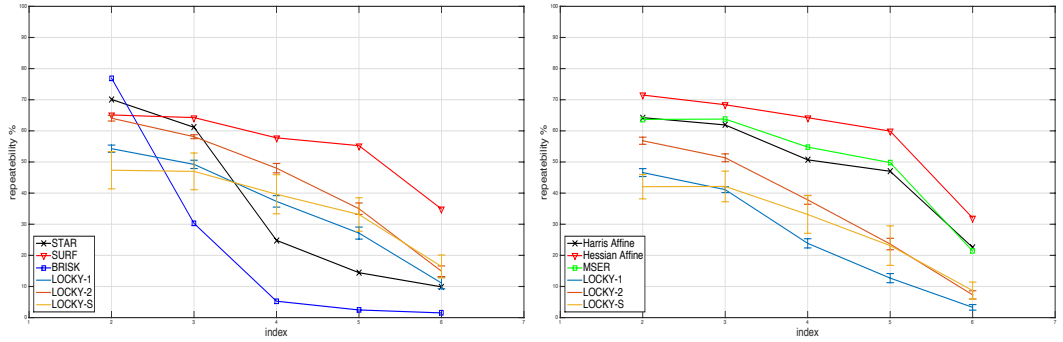
D is the normalised variance across all C_b's, thus the smaller the value of D the more sparse are the features. Capturing the same information, this measure is in some sense an 'inverse' of the entropy. Spectral energy is usually not evenly distributed within an image, this happens more commonly in natural scenarios. When features cluster in certain regions of an image, it can be understood that locality is not successfully achieved. That is, more features tend to be detected in regions with higher spectral energy; this could be the result of normalisation or thresholding processes, as a consequence, features in regions with less energy are lost. Therefore, small values of the dispersion index indicate that features are detected independently from the spectral energy distribution.



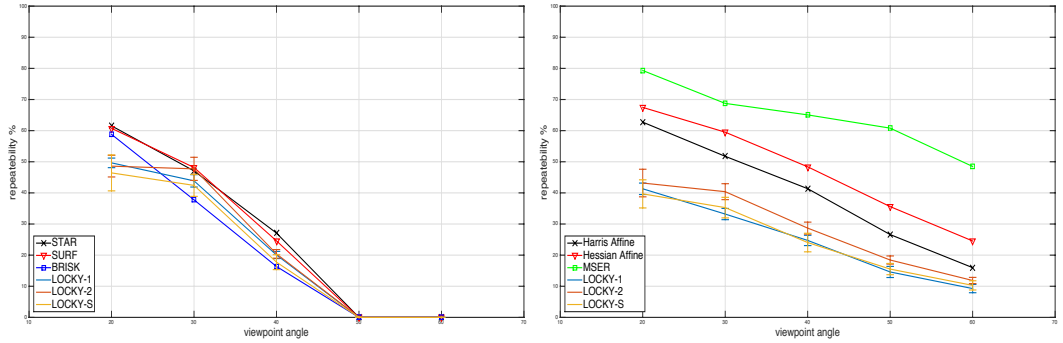
(a) Bark sequence



(b) Bikes sequence

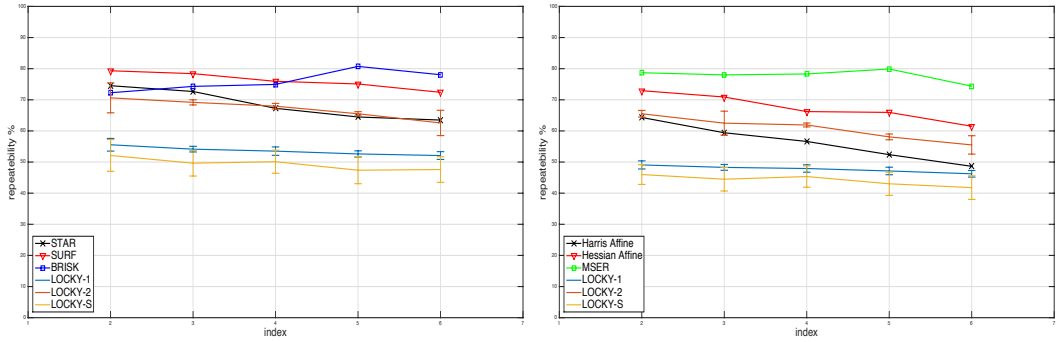


(c) Boat sequence

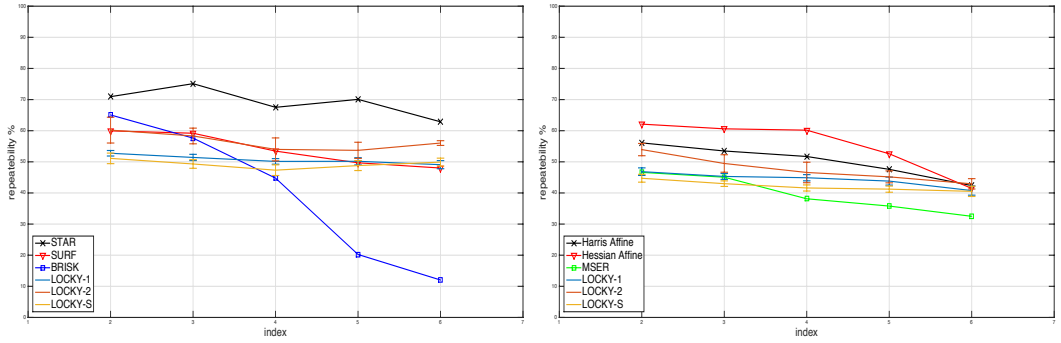


(d) Graffiti sequence

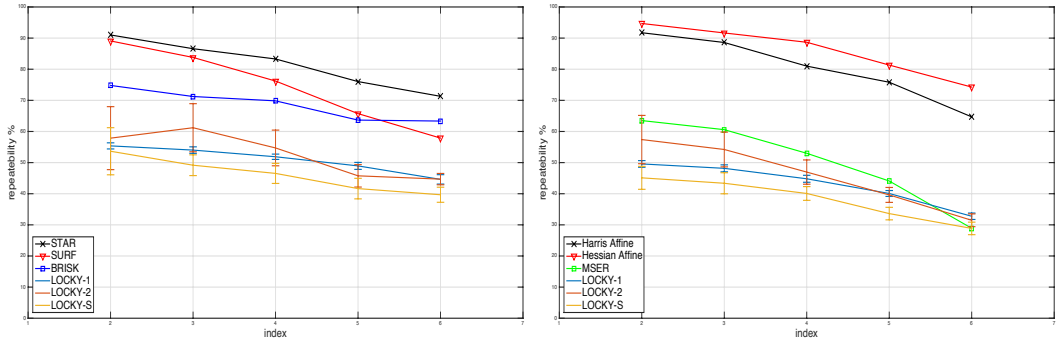
Figure 4.9: The repeatability test presented in [75]. Figures on the left column present the results of the repeatability test with no affine-covariance (features are circles); the right column shows the figures with the results using affine-covariance (features are ellipses). LOCKY-1 uses 1×10^5 rectangles of size ranging from 2^3 to 2^5 and a threshold of 12%; LOCKY-2 uses 1×10^6 rectangles of the same characteristics.



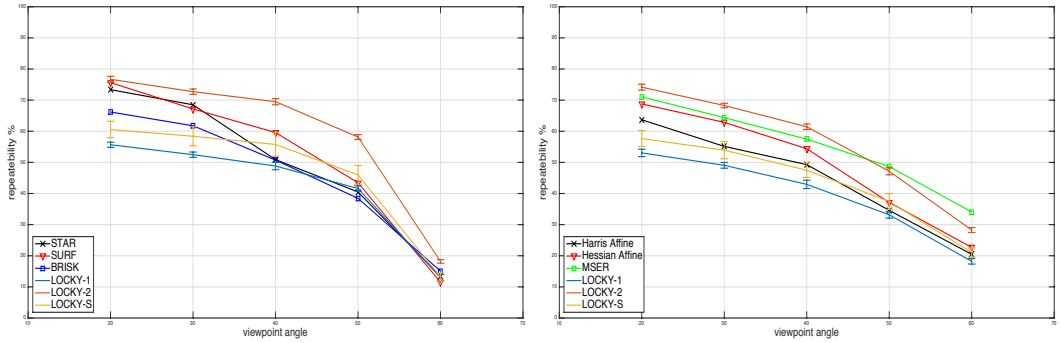
(e) Leuven sequence



(f) Trees sequence



(g) UBC sequence



(h) Wall sequence

Figure 4.9: Continued.

4.4 Conclusion

A new algorithm was presented that (via a novel non-deterministic low level operator), can detect blobs and extract information about image transformations including affine changes in very small amounts of time. The most time consuming step is the BCT, memory accesses and operations are independent through each vote, making the BCT suitable for parallelisation. From the results we conclude that non-deterministic algorithms can help accelerate the search through different scale spaces.

LOCKY and LOCKY-S provide a good alternative for fast feature detection, timing results show that the presented algorithm is amongst the fastest feature detectors available and provides a trade-off between run-time and performance. On average, the LOCKY approach can deliver good performance: competes with those techniques already available and with its own advantages, namely speed and sparseness. LOCKY detector might appear better on rectilinear structures (the wall) than on objects (the boat), this can be investigated further.

While LOCKY uses only rectangles as the shape for the search of blobs, the use of different polygons (using rotated integral images) can be an advantage in terms of precision on the detection although it could cause an increment in the running time. The detection of ridges is shown as a potential application and can also be extended and used for feature detection. A texture is a repetitive pattern, therefore, two votes will progress in a similar manner if both are wholly contained in a single texture; the BCT can be extended for fast analysis of textures.

Features detected using the BCT-S tend to be more congruent with what we recognise as individual objects while those detected with the BCT are smaller (see the image of oranges in figure 4.4). Timings and sparseness results indicate that LOCKY is a good alternative for several applications including landmark selection and image matching.

Chapter 5

Spectral analysis of image-patches

Computer vision has taken great advantage from the development of powerful computers. More complex and computationally demanding calculations are now possible to process thus, computers can interpret large amounts of data to find repetitive patterns of different forms. The field of Convolutional Neural Networks (CNNs) has benefited from this fact, and applications using such methods seem to constantly outperform those that do not. Many aspects of the CNN framework remain open for discussion, one of them is their relation to the detection of features.

Many state-of-the-art implementations in areas like pose estimation, image retrieval, texture description, keypoint prediction, visual content analysis and scene labelling [116, 20, 64, 10, 29], still use detectors like MSER, Difference of Gaussians, Hessian-affine or, a simple dense sampling of the image [72, 74, 60]. Feature detection is a popular open area of research with new algorithms being constantly presented [86, 101].

The objective of feature detection is to find repeatable features disregarding image transformations such as scale, rotation, brightness or noise. Some applications are able to relax some assumptions thus, invariance to some of these transformations can be obviated. For example, Richardson and Olson present a stereo visual odometry system that uses machine learning to construct convolutional filters [86]. Features detected with these filters are not rotation invariant.

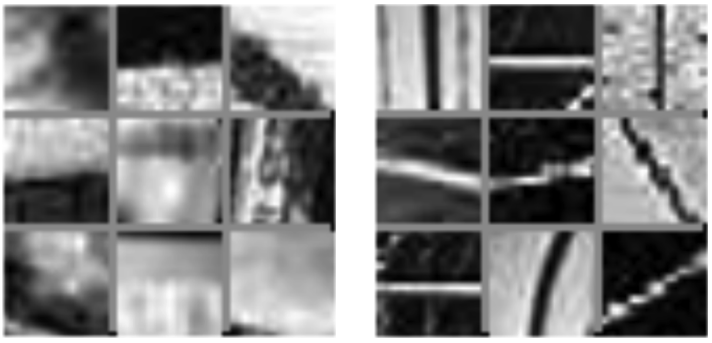


Figure 5.1: Edge-like structures detected with rotation-invariance. Feature models are learnt using a clustering algorithm on the magnitude of the BF moments.

Recently, algorithms for visual content analysis and image retrieval have started building global descriptors from the neural activations of a CNN [7, 58]. This is in a certain

manner, encoding detection and description in a single step. There are then three general ideas for feature detection: the classic feature detection scheme (corners, edges, blobs, etc.) which provides rotation invariance, the CNN approach, and the naive dense sampling of an image.

While classic feature detection algorithms find specific hand-crafted structures, CNN training methods (also known as representation learning) take large sets of images to find patterns that persist throughout the data. Since Hinton *et al.* [40] introduced the idea of representation learning, several algorithms have been presented for such dimensionality reduction. Some of these algorithms impose orthogonality amongst feature representations (PCA), and some others maximise sparseness allowing models to be correlated while maintaining distinctiveness.

As discussed by Bengio *et al.* [15], this area has developed two lines. The first one is based on the neural network model, in which auto-encoders are used to directly learn repetitive structures. The second is based on building *probabilistic models*, local-maxima in the probability density function of the data are model structures. Methods with the neural network model use supervised learning to train deep architectures all layers at once. In contrast, the latter approach uses unsupervised learning to learn patterns one layer at a time.

The scheme for unsupervised representation learning can be depicted as a three step process: the first step is the random extraction of image patches from unlabelled data, the second step is to pre-process the information and, use an unsupervised learning algorithm to learn the representations as a last step. When these representations are learnt directly from images *i.e.* the first layer which maps the image into a feature space, the result is a set of translated and rotated filters that resemble Gabor filters [21].

In this chapter, we introduce a novel feature detection technique that lets the computer learn rotation-invariant model structures using a probabilistic model. This scheme allows the statistical advantage of representation learning to be used as a generic feature detector. As a pre-processing stage, patches are mapped into a rotation-invariant space using the image moments framework. The *Bessel-Fourier* (BF) moments are proven to capture detailed information, outperforming other moment generating functions [108]. The magnitude of the BF moments is rotation-invariant, this property is used to take patches from the pixel domain to a rotation-invariant space where visually similar structures are close to each other.

We analyse the magnitude of the BF moments in terms of the symmetry of features and categorise them into three symmetry groups: non-symmetric, symmetric and composite-symmetric. This new categorisation spans a wide range of structures, allowing for more flexibility than the usual categorisation *i.e.* corners, edges, blobs, etc.

To learn the representations, patches in the rotation-invariant space are clustered using K-means. The resulting learned centroids are edge-like structures in a rotation-invariant space (figure 5.1). Using a non-linear distance measure, patches are compared against one or many of the centroids to create feature-maps.

The abundance of detected features is a key factor for many applications, some techniques are forced to use different detectors [116], detect features on transformed versions of the image [52], or use weaker responses. This approach allows for the creation of an arbitrary number of feature-maps with a single technique. One can pick different structures to be recognised as features when a denser sampling of the image is needed, detecting a rough average estimate of 1500 features per map. The repeatability test presented by Mikolajczyk *et al.* [75] measures persistence of detected features across image transformations, the method presented here shows very competitive scores on such test compared to other state-of-the-art approaches.

The outline of this chapter is as follows. The image moments framework is introduced, the generalized complex moments generating function is derived from the Fourier transform of a 2D function in cylindrical coordinates. An analysis on the symmetry of features in terms of the magnitude and phase of their BF moments is shown. The process for building the rotation-invariant model structures is then presented. Repeatability results are shown, the new approach is compared against other state-of-the-art feature detectors including LOCKY (chapter 4). We close with a small discussion on the new approach.

5.1 Image Moments

In mathematics and in physics, a moment is a measure that combines a physical quantity (*e.g.* electric charge, mass, probability density, brightness, etc.) and a distance. The n^{th} order moment of a punctual particle, and thus the simplest representation of a moment is

$$\zeta_p = x^p f, \tag{5.1}$$

where f is the physical quantity being measured and x^p is the distance function (also known as kernel, moment generating function or moment function). When dealing with particles in higher dimensional spaces, the physical quantity can take the form of a function:

$$\zeta_p = \int_{-\infty}^{\infty} x^p f(x) dx. \tag{5.2}$$

In Computer Vision, the study of images using this framework is known as *image moment analysis* or just image moments. The physical quantity being measured is the brightness of pixels and the mathematical analysis is bounded to the space of the image.

Image moments are coefficients of a generalised Fourier series using different basis functions *e.g.* compare equations 5.2 and 5.5. Using complete sets of orthogonal basis functions allows a complete description of an image, because of this, information about its structure can be extracted. A Fourier series is the decomposition of a function as a linear combination of orthogonal basis functions; for example, the basis functions of the Fourier series in equation 5.3 are sine and cosine waves.

$$f(x) = \sum_{m=-\infty}^{\infty} A_m e^{imx}. \tag{5.3}$$

In a generalised manner,

$$f(x) = \int_{-\infty}^{\infty} e^{2\pi i \upsilon x} F(\upsilon) d\upsilon. \tag{5.4}$$

Solving for $F(\upsilon)$,

$$F(\upsilon) = \int_{-\infty}^{\infty} e^{-2\pi i \upsilon x} f(x) dx. \tag{5.5}$$

Equation 5.5 is the one-dimensional Fourier transform, it is a function describing the coefficients of a Fourier series also known as the *power spectrum*.

Research in image moments is well developed; since Hu published his paper *Visual Pattern Recognition by Moment Invariants* in 1962, the approach appealed to the community finding applications such as pattern recognition, orientation estimation and image compression amongst others [41]. The field of image moment analysis has developed much, since Hu's moment invariants derived from *geometric moments*, many generalisations and alternative moment functions have been researched aiming to find a more compact and complete representation of images.

5.1.1 Geometric moments

Mukundan shows a derivation of the geometric moments from the Fourier transform [77]. In their simplest representation, image moments are the two-dimensional version of equation 5.2 bounded to the region I where the image $f(x, y)$ is defined. These moment kernels are also known as *Cartesian moments* or *regular moments*.

$$\zeta_{p,q} = \iint_I x^p y^q f(x, y) dx dy. \tag{5.6}$$

Equation 5.6 shows how to calculate the $(p + q)^{th}$ order geometric moment. The distance function $x^p y^q$ is combined with the intensity image function $f(x, y)$ to calculate the moments. Transformations such as translation, rotation or changes in scale have a direct influence over the measure. For some applications this fact is irrelevant; this

is not the case for many others that involve pattern recognition, where invariance to transformations is needed.

5.1.2 Central moments and moment invariants

The interpretation of image moments can be deduced from equation 5.6. For instance, the zero-th order moment $\zeta_{0,0}$ is merely the integral of the intensities of the image; the first order moments $\zeta_{1,0}$ and $\zeta_{0,1}$ represent a linear balance around the x and y axes. The intensity centroid of the image (x_0, y_0) (the mean) can be calculated with equation 5.7.

$$\begin{bmatrix} x_0 \\ y_0 \end{bmatrix} = \frac{1}{\zeta_{0,0}} \begin{bmatrix} \zeta_{1,0} \\ \zeta_{0,1} \end{bmatrix}. \tag{5.7}$$

Changes in location can be accounted for by subtracting the location of the intensity centroid as shown in equation 5.8.

$$\mu_{p,q} = \iint_I (x - x_0)^p (y - y_0)^q f(x, y) dx dy. \tag{5.8}$$

These location-normalised moments are known as *central moments*. From equation 5.7 and 5.8, $\mu_{0,0} = \zeta_{0,0}$ and $\mu_{1,0} = \mu_{0,1} = 0$. The second order central moments represent the variance, similarly central moments can be normalised with respect to scale with equation 5.9. Other scale-normalisation schemes exist [14, 83].

$$\eta_{p,q} = \frac{\mu_{p,q}}{\mu_{0,0}^{(p+q+2)/2}}. \tag{5.9}$$

Moment rotation invariants impose a slightly more complicated problem. Given the rotation matrix,

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) \\ sin(\theta) & cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}. \tag{5.10}$$

By linearity, rotated moments can be directly obtained; for example,

$$\begin{bmatrix} \zeta'_{2,0} \\ \zeta'_{1,1} \\ \zeta'_{0,2} \end{bmatrix} = \begin{bmatrix} \left(\frac{1+cos(2\theta)}{2}\right)\zeta_{2,0} - sin(2\theta)\zeta_{1,1} + \left(\frac{1-cos(2\theta)}{2}\right)\zeta_{0,2} \\ \left(\frac{sin(2\theta)}{2}\right)\zeta_{2,0} + cos(2\theta)\zeta_{1,1} - \left(\frac{sin(2\theta)}{2}\right)\zeta_{0,2} \\ \left(\frac{1-cos(2\theta)}{2}\right)\zeta_{2,0} + sin(2\theta)\zeta_{1,1} + \left(\frac{1+cos(2\theta)}{2}\right)\zeta_{0,2} \end{bmatrix}. \tag{5.11}$$

From equation 5.11, $\zeta'_{2,0} + \zeta'_{0,2} = \zeta_{2,0} + \zeta_{0,2}$ which is a term independent from θ and is therefore rotation invariant.

5.1.3 Complex moments

In general, all image moments that use complex kernels are referred to as *complex moments*. In a more particular scene, the *complex radial moments* are functions represented in polar coordinates in the form of equation 5.12. From here on we will refer to complex radial moments simply as complex moments.

$$\psi_{m,n} = \int_0^{\infty} \int_0^{2\pi} \Upsilon_{m,n}(r) e^{-im\theta} f(r,\theta) r d\theta dr. \tag{5.12}$$

The magnitude of complex moments is rotation invariant as the operation creates isotropic kernels; on the contrary, their phase is rotation covariant. The exponential term in equation 5.12 creates sinusoidal waves around the origin in the direction of θ whilst equation 5.13, creates them in the x and y directions.

The major advantage of complex moments over geometric moments is that, their polar nature makes them a useful tool to obtain information about rotational characteristics of images; on the other hand, there exists a loss of information during the discretisation process.

5.1.4 Bessel-Fourier moments

Consider the two-dimensional continuous function $f(x,y)$ and its Fourier transform

$$F(\upsilon,\nu) = \iint_{-\infty}^{\infty} e^{-2\pi i(\upsilon x + \nu y)} f(x,y) dx dy. \tag{5.13}$$

$$\begin{aligned} x &= r cos(\theta), & \upsilon &= \rho cos(\gamma), \\ y &= r sin(\theta), & \nu &= \rho sin(\gamma), \\ r &= \sqrt{x^2+y^2}, & \rho &= \sqrt{\upsilon^2+\nu^2}, \\ \theta &= arctan(y/x), & \gamma &= arctan(\nu/\upsilon). \end{aligned} \tag{5.14}$$

Equation 5.15 is a *multi-pole expansion* of an image in cylindrical coordinates.

$$f(r,\theta) = \sum_{m=-\infty}^{\infty} f_m(r) e^{im\theta}, \tag{5.15}$$

where,

$$f_m(r) = \frac{1}{2\pi} \int_0^{2\pi} f(r,\theta) e^{-im\theta} d\theta. \tag{5.16}$$

Substituting equation 5.15 in 5.13 we obtain the intermediate form,

$$F(\upsilon,\nu) = \iint_{-\infty}^{\infty} e^{-2\pi i(\upsilon x+\nu y)} \sum_{m=-\infty}^{\infty} f_m(r)e^{im\theta}dxdy. \tag{5.17}$$

Using the relations in equation 5.14 the cylindrical form of the Fourier transform is,

$$F(\rho,\gamma) = \int_0^{\infty}\int_{-\pi}^{\pi} e^{-2\pi ir\rho cos(\theta-\gamma)} \sum_{m=-\infty}^{\infty} f_m(r)e^{im\theta}rdrd\theta. \tag{5.18}$$

Changing the order of summation and making $\phi = \theta - \gamma$,

$$F(\rho,\gamma) = \sum_{m=-\infty}^{\infty} \int_0^{\infty} f_m(r)e^{im\gamma}\left[\int_{-\pi}^{\pi} e^{-2\pi ir\rho cos(\phi)}e^{im\phi}d\phi\right]rdr. \tag{5.19}$$

It is pertinent now to introduce the Bessel functions. These functions are the solutions $y(x)$ to Bessel's differential equation,

$$x^2\frac{d^2y}{dx^2} + x\frac{dy}{dx} + (x^2-\alpha^2)y = 0. \tag{5.20}$$

There exist many representations for the Bessel equations; we use the integral form of the Bessel functions of the first kind (figure 5.2),

$$J_m(2\pi r\rho) = \frac{i^m}{2\pi}\int_{-\pi}^{\pi} e^{-2\pi ir\rho cos(\phi)}e^{im\phi}d\phi. \tag{5.21}$$

Figure 5.2: Bessel functions of the first kind for $m = 0, 1, 2, 3, 4$.

Bessel functions are arguably the most frequently used higher transcendental functions [11]. When α is integer, these functions are also known as *cylindrical harmonics*. These equations are importantly related to differential equations of potential, wave motion or diffusion in cylindrical or spherical coordinates.

Substituting equation 5.21 in 5.19,

$$F(\rho,\gamma) = 2\pi \sum_{m=-\infty}^{\infty} (-i)^m e^{im\gamma} \Big[\int_0^{\infty} f_m(r) J_m(2\pi r\rho) r dr \Big]; \tag{5.22}$$

and making,

$$H_m(\rho) = 2\pi \int_0^{\infty} f_m(r) J_m(2\pi r\rho) r dr, \tag{5.23}$$

$$F_m(\rho) = (-i)^m H_m(\rho), \tag{5.24}$$

$$F(\rho,\gamma) = \sum_{m=-\infty}^{\infty} F_m(\rho) e^{im\gamma}. \tag{5.25}$$

The function $H_m(\rho)$ is known as the *Hankel transform*, this function describes the expanding waves around the origin. Equation 5.25 shows the Fourier transform in cylindrical coordinates. This is a complete representation of the function $f(r,\theta)$ with $F_m(\rho)$ being its power spectrum. The variable ρ represents the expanding behaviour of the function, and γ describes its rotational characteristics.

If we substitute equation 5.16 in equation 5.23 we obtain,

$$H_m(\rho) = \int_0^{2\pi} \int_0^{\infty} J_m(2\pi r\rho) e^{-im\theta} f(r,\theta) r dr d\theta. \tag{5.26}$$

Choosing $\rho = {}^{\lambda_{m,n}}/_{2\pi}$,

$$B_{m,n} = \int_0^{2\pi} \int_0^{\infty} J_m(\lambda_{m,n} r) e^{-im\theta} f(r,\theta) r dr d\theta. \tag{5.27}$$

Where $\lambda_{m,n}$ is the n^{th} zero of the m^{th} order Bessel function of the first kind. The set of moments in equation 5.27 is a special case of the definition of complex moments in equation 5.12, where

$$\Upsilon_{m,n}(r) = J_m(\lambda_{m,n} r). \tag{5.28}$$

Equation 5.27 is a more specific representation of the BF moments as presented in [109]. Equation 5.25 is continuous in ρ. The rotational repetition m has to be a natural number, otherwise the term $J_m(\lambda_{m,n} r) e^{-im\theta}$ would be discontinuous. The set of all $B_{m,n}$ are the coefficients of the Fourier-Bessel series in equation 5.25. The real part of the intrinsic kernel of some BF moments are shown in figure 5.3.

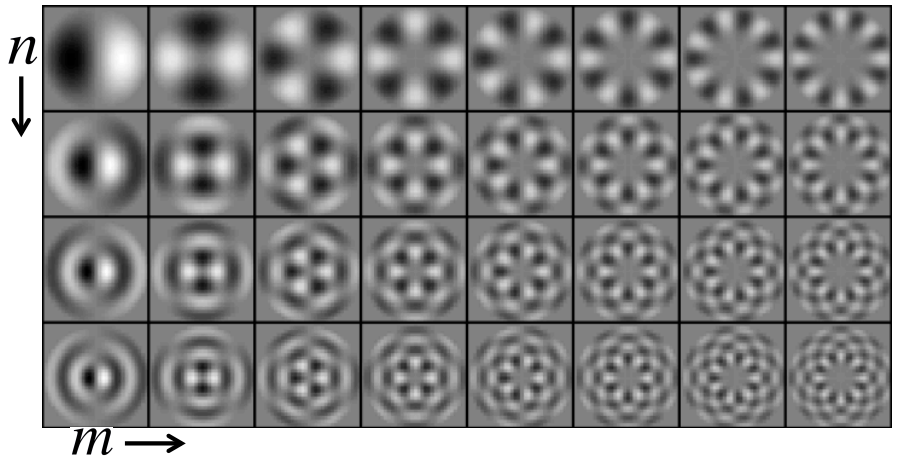


Figure 5.3: Real part of the kernels of the BF moments for $m = 1, 2, \ldots, 8$ (rows), and $n = 0, 1, 2, 3$ (columns).

5.1.5 Other image moment functions

To minimise the redundancy of information every moment contains, it is important to use orthogonal sets of functions. The set of BF moments is both complete and orthogonal. Of course, this is not the only complete set of orthogonal moments; for instance, the Legendre moments [95] are a set defined in Cartesian coordinates. The Zernike moments [95] are a complete set of orthogonal moments where,

$$\Upsilon_{m,n}(r) = R_{m,n}(r) = \sum_{s=0}^{(k-|n|)} (-1)^s \frac{(k-s)!}{s!\left(\frac{k-2s+|n|}{2}\right)!\left(\frac{k-2s-|n|}{2}\right)!} r^{k-2s}, \qquad (5.29)$$

$$k \in \mathbb{N}; \quad 0 \leq |n| \leq k; \quad m - |n| \text{ is even.}$$

BF and Zernike moments are orthogonal in the continuous domain, when discretising the functions orthogonality is lost. There are sets of discrete orthogonal moments, some examples of this kind are the Krawtchouk [110], the Chebyshev [76] and the Hahn moments [117]; these moments allow the extraction of more accurate information due to their discrete nature although, they are defined in the Cartesian plane and rotation invariants would have to be constructed.

5.2 The symmetry of features

Interacting in an intricate manner, frequency components are the building-blocks that construct structure and are values difficult to understand. Interpretation of image moments is a long standing subject of discussion in the community. As lower order moments

are simple, higher order moments are difficult to understand. Instead of comparing moments with physical properties, we analyse their meaning in terms of the structures they build together.

It is widely known that some image moments vanish when the image being analysed is symmetric. Some researchers have taken advantage of this property to detect the number of folds in a symmetric image [92]. The absence or presence (zero or non-zero magnitude) of moments, implies that certain frequency components disappear when symmetry is present. Also, phase-congruency characteristics are present in symmetric image-patches. In this section we present the relation between image moments and the symmetry of features. Based on this analysis, symmetrical image-patches can be created. We categorise image-patches into three symmetry-groups based on different characteristics of their power spectrum.

5.2.1 Non-periodic 2D symmetries

Many characterisations of symmetry exist, Weyl provides a good basis for the study of mathematical and computational symmetry [106]. The description of symmetry that Dickinson proposes, is used to extract information from images and ease the understanding of shape [26]. Griffin presented a compact characterisation of the possible non-periodic image symmetries, these are shown in table 5.1 [34]. Here we use the notation S instead of J to avoid confusion with the notation of the Bessel functions.

Group	Description.
S_0	The only automorphisms is the identity transformation.
$S_{1,k}$	k rotations around a common center ($k \geq 2$).
$S_{2,k}$	k mirrors (intersecting at a point if $k > 1$).
S_3	Unchanging in one direction.
S_4	Unchanging in one direction with a line of reflection in the same direction.
S_5	Infinite rotations around a point ($S_5 = S_{1,\infty}$).
$S_{6,k}$	k anti-mirrors (intersecting at a point if $k > 1$).
$S_{7,2k}$	k anti-rotations and k rotations around a point.
$S_{8,2k}$	k anti-mirrors and k mirrors intersecting at a point.
S_9	Unchanging in one direction with a line of anti-reflection in the same direction.
S_{10}	The image can be translated in one direction with a matching proportional change in intensity.
S_{11}	S_{10} with a line of reflection in the same direction.
S_{12}	S_{10} with a line of centres of 180 ° anti-rotations in the same direction.
S_{slope}	A non-flat plane.
S_{const}	A constant function.

Table 5.1: Non-periodic symmetries of image-patches as presented in [34].

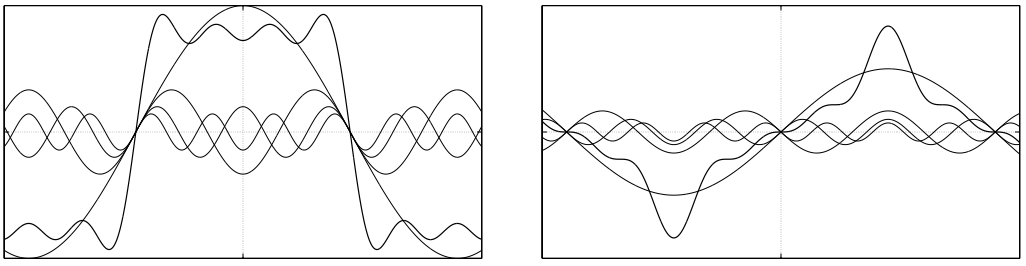
Some of these symmetries have a relation with periodic-symmetries and also amongst each other. $S_{2,k}$ is a special case of $S_{1,k}$ when $k > 1$, S_{10} and S_{11} are also periodic-symmetries and, S_3 and S_4 are also part of $S_{2,k}$, S_{slope} is in $S_{8,2}$, etc.

5.2.2 Phase congruency in cylindrical coordinates

As shown in section 5.1, image-patches can be represented as a linear sum of basis filters. In the case of the BF moments, they represent the spectral density of the image-patch in cylindrical coordinates. Relations between the frequency components of the series create structure, for example, presence or absence of some components may create symmetric patterns. Moreover, alignment in the symmetry axis of moment functions can create symmetrical structures, such alignment is known as phase-congruency.

Kovesi introduced the term *phase-congruency* and used it to detect mirror and anti-mirror symmetry ($S_{2,k}$ and $S_{6,k}$) [47]. Consider the one-dimensional case, odd-symmetric functions such as $sin(t)$ are anti-mirror symmetric with the axis being $t = 0$; if all coefficients a_m equal 0 in the Fourier series in equation 5.30, $f(t)$ will be anti-mirror symmetric at $t = 0$. For the mirror symmetric case, we have to consider even-symmetric functions such as $cos(t)$; in this case, if the coefficients b_m equal 0, $f(t)$ will be mirror symmetric at $t = 0$. Figure 5.4 shows a graphical example of the phase patterns that create symmetry in 1D.

$$f(t) = \frac{a_0}{2} + \sum_{m=1}^{\infty} \Big(a_m cos(mt) + b_m sin(mt) \Big). \tag{5.30}$$



(a) Even-congruent waves creating a mirror symmetric function.

(b) Odd-congruent waves creating an anti-mirror symmetric function.

Figure 5.4: Phase-congruency, these phase patterns create symmetry around $t = 0$.

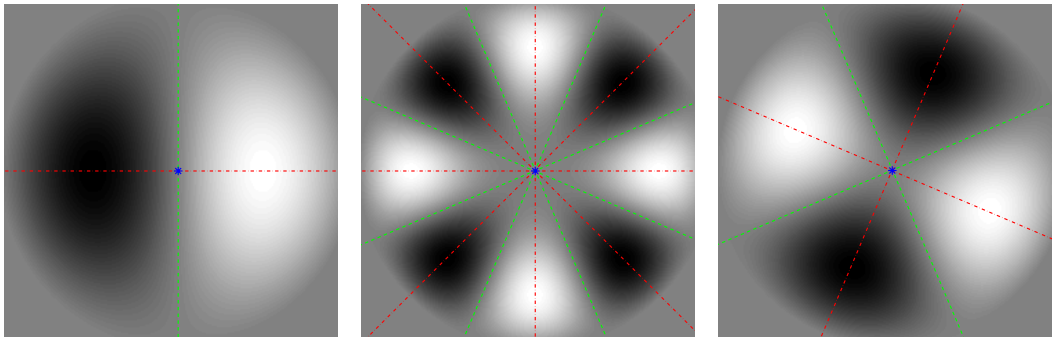
In general, two sinusoidal functions $f(t)$ and $g(t)$ are even-congruent around $t = t_s$ if and only if,

$$\frac{d}{dt} f(t_s) = \frac{d}{dt} g(t_s) = 0. \tag{5.31}$$

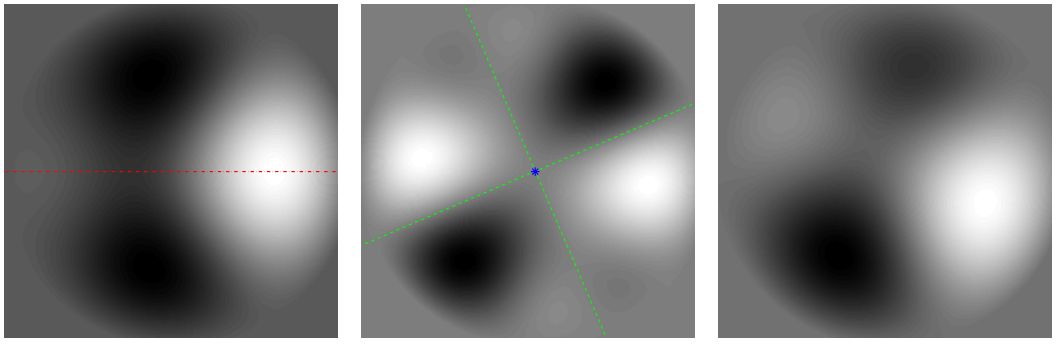
Similarly, two sinusoidal functions are odd-congruent around $t = t_s$ if and only if,

$$f(t_s) = g(t_s) = 0. \tag{5.32}$$

The symmetry is broken if the frequency components are not phase-congruent. Kovesi uses wavelets to calculate local phase; to extend this idea to two dimensions, he uses the same wavelet at the same point in different directions. Phase congruency can be considered as an alignment in the symmetry axis of the frequency components. In a similar fashion, 2D filters in cylindrical coordinates can show phase congruency; if there exists an alignment on the axis of mirror or anti-mirror symmetry, the rotational-frequency components are said to be even or odd phase-congruent. This example is illustrated in figure 5.5.



(a) $re(B_{1,0})$ (b) $re(B_{4,0})$ (c) $re(B_{2,0}) + im(B_{2,0})$



(d) $re(B_{1,0}) + re(B_{4,0})$ (e) $re(B_{4,0}) + re(B_{2,0}) + im(B_{2,0})$ (f) $re(B_{1,0}) + re(B_{2,0}) + im(B_{2,0})$

Figure 5.5: BF moment kernels creating symmetry. The operator $re()$ denotes the real part of the filter, and $im()$ the imaginary part. If a mirror axis (red) of two functions is aligned, the sum of the two functions creates mirror symmetry along the aligned axis (d). In a similar manner if an anti-mirror axis (green) is aligned, the sum of these two functions creates anti-mirror symmetry (e). On the other hand, if no axis is aligned, symmetry will be corrupted (f).

5.2.3 Creating symmetry

There exists an intrinsic relation between the notion of shape, symmetry, and structure in general [26]. Table 5.1 lists all recognised non-periodic image symmetries as presented by Griffin [34]. Such structures are generated by different rotational frequency characteristics, namely the magnitude and phase of BF moments.

It is well known that some moments in the rotational direction (index m in $B_{m,n}$), disappear (have zero magnitude) when there is symmetry present. Consider the rotation

symmetric case $S_{1,k}$, moments have non-zero magnitude only when m is an integer multiple of the number of symmetry axes k; in any other case, moments have zero magnitude. Table 5.2 shows the conditions for the rest of the symmetry cases. In accordance with Kovesi, we distinguish *even* or *odd phase congruency* as an alignment between the axes of symmetry or anti-symmetry of two or more functions. It is possible to depict three major groups of structures (A fourth group and an exception are discussed in section 5.6, synthetically generated images are shown in figure 5.6):

- The *non-symmetric group* is characterised by S_0 where all $B_{m,n}$ may have non-zero magnitude.
- The *symmetric group* encloses $S_{1,k}$, $S_{2,k}$ and $S_{6,k}$, $B_{m,n}$ may have non-zero magnitude when $m = ak$ $\{a \in \mathbb{N}, k > 1\}$. For instance $S_{1,3}$ in figure 5.6, in this case $k = 3$, therefore $B_{m,n}$ may take non-zero magnitude when $m = 3, 6, 9, etc.$
- The *composite-symmetric group* with $S_{7,2k}$ and $S_{8,2k}$, moments $B_{m,n}$ in this group may have non-zero magnitude when $m = (2a + 1)k$ $\{a \in \mathbb{N}\}$. Consider $S_{8,2}$ in figure 5.6, here $k = 1$ hence, moments may take non-zero magnitude when $m = 1, 3, 5, etc.$

Group	Description.
S_0	At least two frequency components with no common prime factors, required not to be phase-congruent.
$S_{1,k}$	All integer multiples of k, required not to be phase-congruent.
$S_{2,k}$	All integer multiples of k, required to be even-congruent.
S_3	$S_{2,1}$.
S_4	$S_{2,2}$.
S_5	Zero frequency components.
$S_{6,k}$	All integer multiples of k, required to be odd-congruent.
$S_{7,2k}$	Odd multiples of k. Both types of congruency at the same angle are required.
$S_{8,2k}$	Odd multiples of k, odd-congruency required.
S_9	$S_{8,2}$.
S_{slope}	$S_{8,2}$.
S_{const}	All frequency components which intrinsic kernel does not integrate to zero.

Table 5.2: Existence and phase conditions for image moment functions to create symmetric patterns.

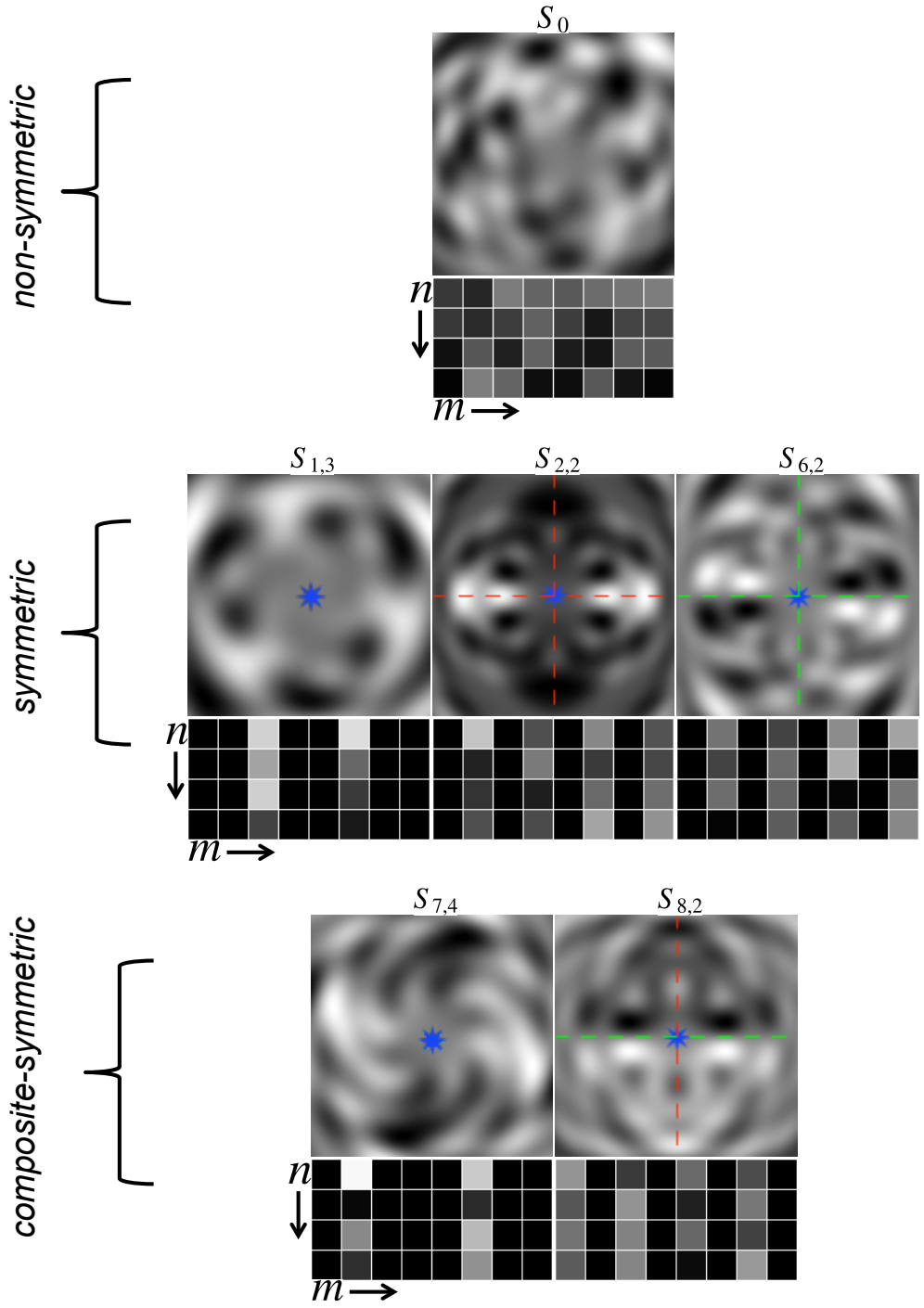


Figure 5.6: Synthetically generated symmetry. Different symmetries reconstructed from randomly generated $B_{m,n}$ moments complying with the conditions stated in table 5.2. Axes of mirror and anti-mirror symmetry in red and green respectively, a blue star denotes the centre of rotation symmetry. Beneath, the magnitude of the BF moments used to generate each image for $m = 1, 2, \ldots, 8$, and $n = 0, 1, 2, 3$ (normalised to black = 0, and white = 1). Three distinct groups of symmetry characterised by non-zero magnitude of certain moments are recognised.

5.3 Unsupervised learning to find structures in a rotation invariant space.

Finding the analytical solution to $B_{m,n}$ for any given function is mathematically complex and requires deep knowledge on the matter (see section 5.3.3). Using the idea for unsupervised representation learning, the manual selection of structures can be obviated by clustering samples in a rotation-invariant space created from the magnitude of the BF moments.

The resulting centroids of this process are used as feature models to compare patches in an image, allowing the detection of arbitrary shapes as features. The feature models are generally rotation-invariant representations of edge-like structures (figure 5.1). The detected features show very competitive results on the repeatability test presented in [75] compared to other state-of-the-art detectors. Moreover this approach can recognise multiple structures as features thus, the user can substantially increase the amount of detected features in a controlled manner.

5.3.1 A rotation-invariant space

The magnitude of the BF moments provides a rich rotation-invariant descriptor that may be used for comparison with other patches. When $m = 0$, BF moments are not invariant to changes in brightness; we use $\{m = 1, 2, \ldots, M;\ n = 0, 1, \ldots, N-1\}$. The complex magnitude of the $B_{m,n}$ is used to map a patch into an $M \times N$-dimensional vector noted as $\tilde{\mathbf{d}}_x$.

The elements of the vector $\tilde{\mathbf{d}}_x$ are the magnitude of the coefficients of the FT in cylindrical coordinates. This vector is therefore the power spectrum in the region bounded by the variables M and N. The energy of the patch in this region of the spectrum does not vary with changes in brightness, on the other hand, changes in contrast affect the energy of the signal. To avoid these variations, vectors $\tilde{\mathbf{d}}_x$ are normalised using equation 5.33.

$$\mathbf{d}_x = \frac{\tilde{\mathbf{d}}_x}{||\tilde{\mathbf{d}}_x||}. \tag{5.33}$$

Where $\tilde{\mathbf{d}}_x$ are the un-normalised vectors. In practice, we set $\mathbf{d}_x = \mathbf{0}$ (all elements equal 0) when $||\tilde{\mathbf{d}}_x|| < E_{thresh}$ to avoid highlighting regions of low energy or division by 0 in its case. In a general sense, the energy of a signal $f(t)$ is

$$E = \int_{-\infty}^{\infty} |f(t)|^2 dt. \tag{5.34}$$

Equation 5.35 shows Parseval's theorem for one dimension, it extends to multiple dimensions and means that the energy of the signal is preserved throughout the FT *i.e.* it is a unitary transform. In a certain manner, normalising the magnitude of the Fourier coefficients is equivalent to normalising the variance of the patch.

$$\int_{-\infty}^{\infty} |f(t)|^2 dt = \int_{-\infty}^{\infty} |F(\upsilon)|^2 d\upsilon. \tag{5.35}$$

In the frequency domain, the energy of a signal within a range of the spectrum can be calculated by reducing the integration range. The vectors $\tilde{\mathbf{d}}_x$ contain the power-spectrum of the image patches, therefore the energy of the image patch in the range delimited by M and N is

$$E_d = \sum_{m,n} |B_{m,n}|^2 = ||\tilde{\mathbf{d}}_x||^2. \tag{5.36}$$

The variable E_{thresh} represents the minimum acceptable energy, in the region bounded by M and N, for an image-patch to be considered. Because of this, E_{thresh} is highly dependent on the variables M and N; as these values increase E_{thresh} should increase as well. The mathematics for analytically calculating this value are complex as they involve integrating Bessel functions (see section 5.3.3). For practical implementation we opt for empirically chosen values that return an approximate desired number of detections.

The vectors $\mathbf{d}_x$ are non-negative unit-vectors (*i.e.* $||\mathbf{d}_x|| = 1$ with all non-negative dimensions). Hence, the normalisation process maps these vectors on a *Quadrantal Spherical Triangle* (figure 5.9). Patches that are visually similar are close to each other in this space.

5.3.2 An approach to detect low-level symmetry

Normalising the energy in a closed range lets us understand the rotation invariant space in more depth. Convolving an image with eight $B_{m,n}$ kernels with $m = 1, 2, \ldots, 8$ and $n = 0$, each pixel is represented by a vector $\tilde{\mathbf{d}}_x$ of eight elements. After the normalisation process, the vector $\mathbf{d}_x$ of each pixel brings out the energy distribution in that region of the image. This process can be seen in figure 5.8, at the bottom image after the normalisation process the eight-fold rotational symmetric patches are highlighted.

Using non-maxima suppression on the normalised maps, it is possible to detect features of the desired amount of rotational repetitions; this is shown in figure 5.7, where eight-fold (top) and five-fold (bottom) rotational patches are detected. This process is simple but not very useful as not many real-world images show high-quality rotational symmetric features. Although the normalisation is performed within a region of the spectrum, the detection relies only on the dominance of energy at a certain frequency (only one element of the vector $\mathbf{d}_x$).

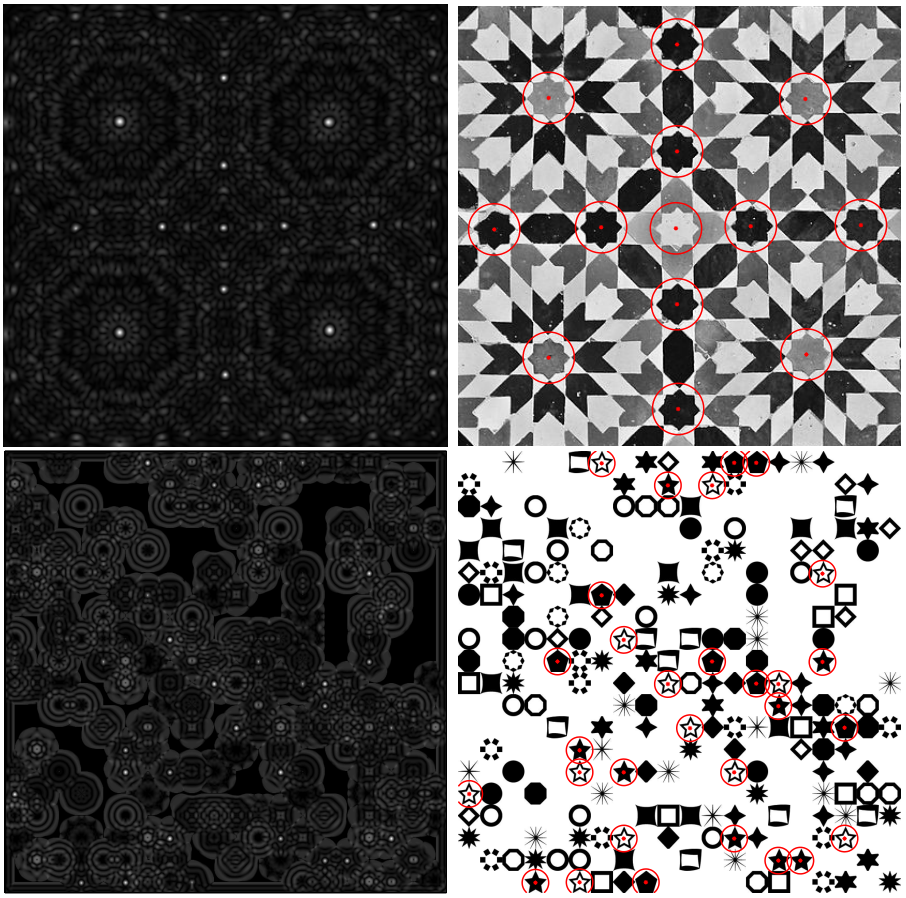


Figure 5.7: Detecting symmetric image-patches using the presented framework.

5.3.3 Manual selection of feature models

In section 5.2, it was shown that the interactions between BF moments are the building blocks of structure. The process shown in section 5.3.2 does not take this into account. To be able to find more complex structures it is necessary to look at the higher-dimensional rotation-invariant space. For completeness we show the BF moment generating function again.

$$B_{m,n} = \int_0^{2\pi} \int_0^{\infty} J_m(\lambda_{m,n} r) e^{-im\theta} f(r,\theta) r dr d\theta. \tag{5.37}$$

The BF moments of a function can be analytically found. For example, a step edge can be written in its polar form as

$$f(r,\theta) = \begin{cases} 1 & 0 \leq \theta < \pi \\ 0 & \pi \leq \theta < 2\pi \end{cases} \qquad \{0 \leq r < 1\}. \tag{5.38}$$

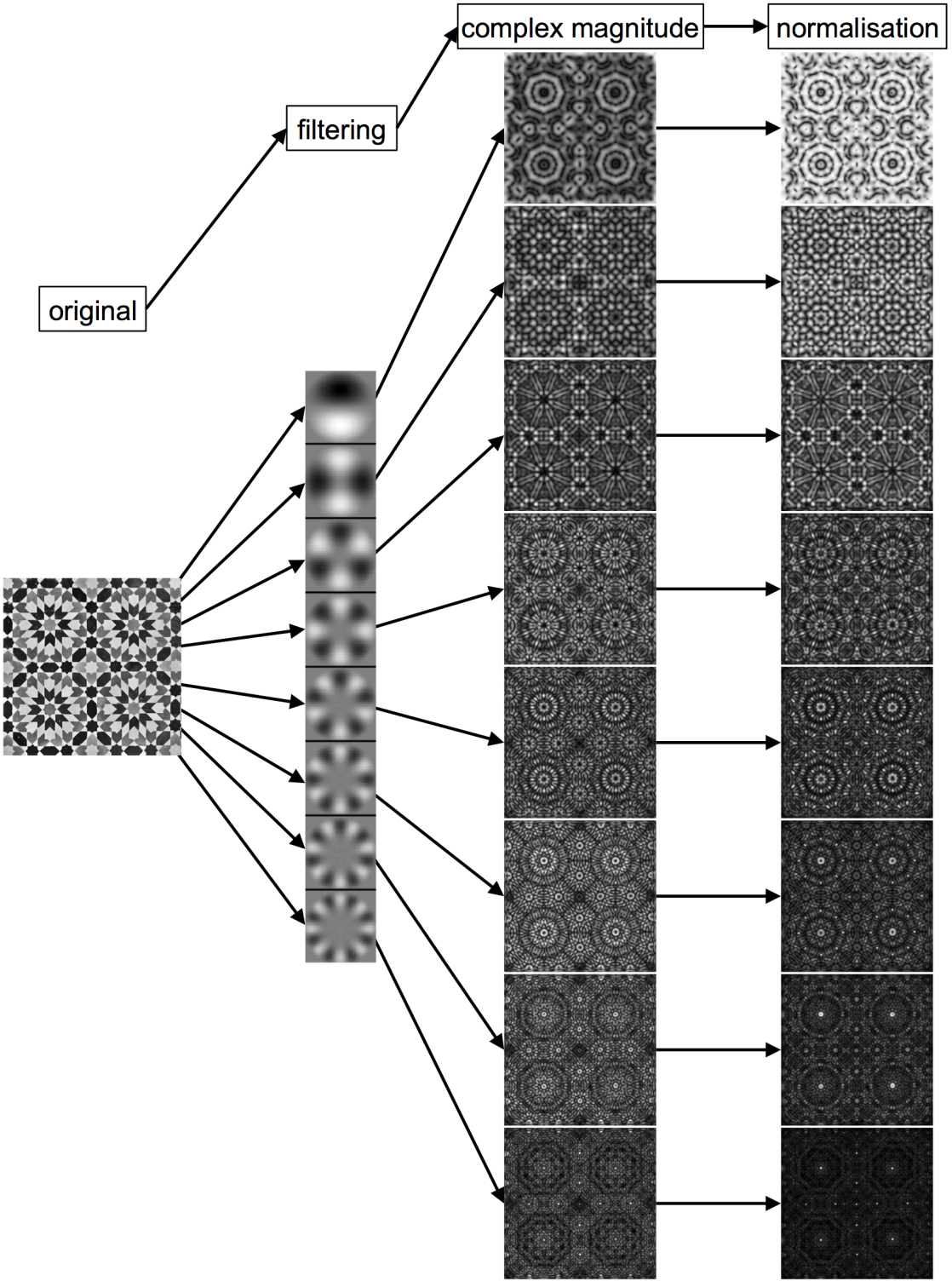


Figure 5.8: The flow for mapping the image into a rotation-invariant space. The image is convolved with a set of BF filters. The complex magnitude of this operation is then normalised.

Substituting equation 5.38 in 5.37 and changing the integration order,

$$B_{m,n} = \int_0^1 r J_m(\lambda_{m,n} r) \Big(\int_0^\pi e^{-im\theta} d\theta \Big) dr, \tag{5.39}$$

$$\int_0^\pi e^{-im\theta} \delta\theta = \begin{cases} 0 & \text{when } m \text{ is even,} \\ -2i/m & \text{when } m \text{ is odd.} \end{cases} \tag{5.40}$$

Substituting equation 5.40 in 5.39,

$$B_{m,n} = \frac{-2i}{m} \int_0^1 r J_m(\lambda_{m,n} r) dr. \tag{5.41}$$

Equation 5.41 holds only when m is odd, for all even values of m, $B_{m,n} = 0$, an edge is a composite-symmetric structure of the type $S_{8,2}$ ($k = 1$). This agrees with the theory presented in section 5.2.3, where it is shown that symmetries of this kind have non-zero moments only when $m = (2a+1)k$ $\{a \in \mathbb{N}\}$ *i.e.* m is odd. The case $m = 0$ will be discussed in section 5.6.

To solve equation 5.41, we can use the Bessel function representation in equation 5.42.

$$J_m(\lambda_{m,n} r) = \sum_{s=0}^{\infty} \frac{(-1)^s}{(m+s)!\, s!} \Big(\frac{\lambda_{m,n} r}{2} \Big)^{m+2s}. \tag{5.42}$$

$$B_{m,n} = \frac{-2i}{m} \sum_{s=0}^{\infty} \frac{(-1)^s}{(m+s)!\, s!} \int_0^1 \Big(\frac{\lambda_{m,n} r}{2} \Big)^{m+2s} r dr. \tag{5.43}$$

And integrating,

$$B_{m,n} = \frac{-i(\lambda_{m,n})^m}{(2)^{m-1}\, m} \sum_{s=0}^{\infty} \frac{(-1)^s\, (\lambda_{m,n})^{2s}}{(2)^{2s}\, (2s+m+1)(m+s)!\, s!}. \tag{5.44}$$

Equation 5.44 decays rapidly with s, therefore a few values are sufficient to derive an appropriate approximation of $B_{m,n}$. In this manner it is possible to analytically find the location of any given structure in the rotation-invariant space. This process is difficult and requires deep understanding on the subject of feature detection. In the next section we show a method to avoid these complex calculations.

5.3.4 Learning repetitive structures

The probability density function (PDF) of the normalised rotation-invariant descriptors unveils persistence of features. Centres of clustering algorithms tend to converge approximately to local-maxima on the PDF of a discrete random variable. These local-maxima represent repeatable features.

K-means is arguably the most well-known clustering algorithm, it is easy to implement and performs well under certain initialisation conditions. As discussed in [81], initialisation of the centres can be an issue, as different strategies lead to different results. Due to the nature of the data we are trying to cluster, the random partition method is arguably the best suited. The process begins by randomly assigning each sample to one of the C clusters, and then calculating the initial location of the centres with the average location of all the members of each cluster. This places the centres near the middle of the distribution and then they move to local-maxima of the PDF.

We cluster the $\mathbf{d}_x$ (section 5.3.1) of 5×10^5 randomly extracted patches from a subset of 500 also randomly selected images from the Caltech101 dataset [30]. If a patch meets the condition $||\tilde{\mathbf{d}}_x|| < E_{thresh}$ it is substituted by another randomly selected patch until this condition is not met. In this application, K-means usually converges after around 8 iterations. The learnt centres are known as *feature models* and take the notation ω_c, where c refers to the index of the learnt centre. For visualisation, a rank-3 approximation of the training data and the learnt centroids is obtained, this is shown in figure 5.9.

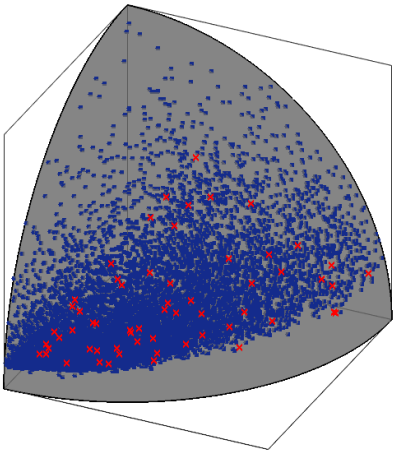


Figure 5.9: A rank-3 approximation on the normalised vectors $\mathbf{d}_x$ in blue and the learnt centres ω_c in red on a unitary quadrantal spherical triangle.

To visualize what the centres ω_c represent, we randomly construct another large set of patches and compare them to the centres using the distance measure in equation 5.45. Figure 5.10 shows some of the closest patches to the centres. Features belonging to the three symmetry groups are presents amongst the learnt structures. For example, features

in the symmetric group can be seen in F1, B8 or F8. Examples of composite-symmetric features are found in C4, C3 or G3. Both types of structures can be recognised in their majority as centred edges of different characteristics. On the other hand, non-symmetric features are present in E8 or H1; these structures are generally present in the form of noisy patches or translated edges. This property was discussed in [21], where it is shown that different feature representation learning methods produce localised filters that resemble Gabor filters.

5.4 Feature detection

Taking advantage of the rotation-invariance provided by the presented method, it is possible to use this scheme as a generic feature detection algorithm. Consider every possible patch of size $w \times w$ on an image, each is mapped into the rotation-invariant space and normalised using the framework presented above. We use now the notation $\mathbf{d}_{x,y}$ to indicate the normalised rotation-invariant vector of the patch with coordinates (x, y) in the image.

Feature detection is usually a similarity measure between a patch and a model structure. To compare the vectors $\mathbf{d}_{x,y}$ to the centres ω_c it is necessary to define a distance measure. Ideally, patches that are visually equal should lie at the same point in the rotation-invariant space. This assumption is a hard restriction as usually images present transformations like noise, blurring, or small changes in perspective. For this reason, our distance measure has to be smooth and decay in a non-linear manner when distancing the feature model. We use the inner product weighted with a Gaussian function shown in equation 5.45 (figure 5.11).

$$D_c(x, y) = e^{-\tau\,(\,\langle \mathbf{d}_{x,y}, \omega_c \rangle - 1)^2}. \tag{5.45}$$

As the magnitude of both vectors $\mathbf{d}_{x,y}$ and ω_c is unitary, the inner product $\langle \mathbf{d}_{x,y}, \omega_c \rangle$ is always in the range $[0, 1]$. The variable τ controls similarity *tolerance*, the bigger the value the lower the tolerance. The value of τ is highly dependent on M and N. This process creates C feature maps $D_c(x, y)$ in which non-maxima suppression can be used to find features. Figure 5.12 shows three feature maps created and the respective features found after non-maxima suppression. Points of interest found in different feature maps are said to be of different class. In this example we fixed the size of the kernels, to detect features at multiple scales we build an image pyramid and detect the features at different octaves.

Parameters for the algorithm must be set to suit the application. The parameters M and N are the amount of moments to be extracted from every patch, in this implementation we use $M = 8$ and $N = 4$; although in our experiments we used various settings, smaller

Figure 5.10: The nine closest patches to each of the learnt centroids. The three symmetry groups are present, the non-symmetric group usually captures noisy patches or translated edges (H7, H8 or D5), the symmetric group has a good response on roof edges of varying width (B1, E2 or B8) and, the composite-symmetric group which tends to highlight step and ramp edges (A4, D2 or E4). The centroids are brightness normalised for easier visualization.

values than $M = 8$ and $N = 1$ appear to be insufficient. As shown in equations 5.34 and 5.35, the variance of a function and its energy are equivalent. If a function is differentiable in a region, its variance in such region is finite and thus its energy. This implies that its spectral energy decays as it approaches infinity, therefore a good approximation can be obtained from a finite number of BF-moments. We choose these values as they appear to provide enough discrimination between features without over-fitting or generating too many noisy feature-models.

The energy threshold E_{thresh} controls the strength of the features to be detected, a small value will trigger weaker responses (see section 5.3.1). In this implementation we use an empirically chosen value $E_{thresh} = 5$. This generates an approximately between 1500 and 2000 features per feature class in a 1024×680 image.

Choosing the number of clusters C is a difficult subject and an area of research on itself [50]. Representation learning algorithms for CNNs usually suggest a value of C "as large as compute resources will allow" [22], learning directly from pixel intensities these features must be abundant enough to cover all rotations of all feature models; our approach covers all rotations in a single model. Due to the feature detection application, this parameter should relate then to the amount of structures to detect as features; to obtain the results shown here, we train the centroids using $C = 64$ and use only 32 of the centroids for the detection (top half of figure 5.10). Training 64 centroids does not seem to over-fit the data, but some of them seem to capture similar information. Randomly choosing 32 of the centroids discards some redundancy whilst keeping a good coverage of information.

The similarity tolerance controlled by τ has a large effect on the distinctiveness among feature classes. Small values of this parameter tend to make feature maps similar to each other. Large values reduce the similarity tolerance and thus repeatability is substantially affected, we set $\tau = 1000$. Added to this parameter a threshold $D_{thresh} = 0.5$ is used in the non-maxima suppression to avoid distorted features, although this parameter could be obviated by fine tuning the value of τ. This value of τ was chosen empirically, a larger value generates too much redundancy in the detected features, a smaller value impairs the detection.

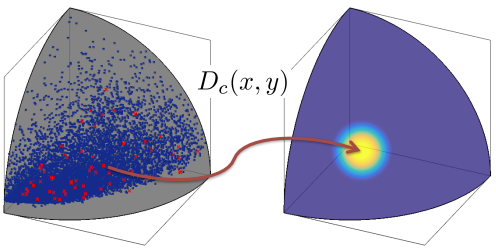


Figure 5.11: The function $D_c(x, y)$ measures the distance in a non-linear manner between the vector $\mathbf{d}_{x,y}$ and the c^{th} feature model.

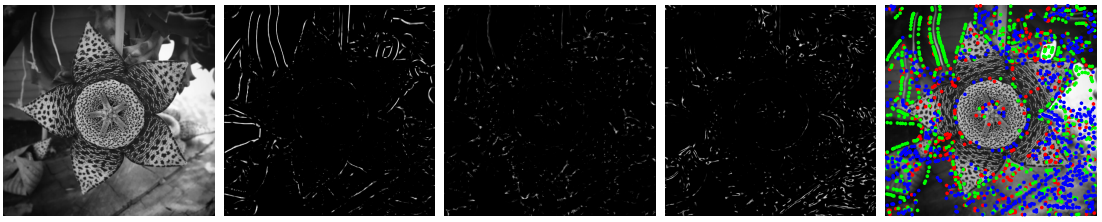


Figure 5.12: Using the learnt centroids we create feature maps $D_c(x, y)$ (three middle images), from an original image (left). Features are detected by applying non-maxima suppression on the maps. Each of the different feature classes is a different colour on the image at the right.

5.4.1 Time complexity analysis

The time complexity of the feature detector is dependent on various parameters. This can be divided into three stages:

- Map every pixel into the rotation-invariant space. This stage of the algorithm performs $2 \times M \times N$ convolutions of size $w \times w$ with non-separable kernels, the reason for the factor 2 is that the kernels are complex. Plus the normalisation process, which is a factor of the dimensionality of the vectors $\mathbf{d}_x$ *i.e.* $M \times N$. The complexity of this stage is $O((\alpha w^2 + \beta)MNX)$, where X is the number of pixels in the image and, α and β are constants.

- Creation of the rotation-invariant feature maps. Once the vectors $\mathbf{d}_x$ have been calculated, each pixel is compared against each of the centres ω_c. This operation is also dependent on the dimensionality of $\mathbf{d}_x$, thus the complexity of this stage is $O(\epsilon MNCX)$ where C is the number of feature maps and ϵ is some constant.

- Non-maxima suppression. These algorithms have a linear complexity on the number of pixels X.

Considering $w \times w$, M, N and C are given constants, the total complexity of the entire process is linear on the number of pixels *i.e.* $O(X)$. For the parameters specified and using 16×16 kernels (figure 5.13), the detection runs in an average of 294 milliseconds per feature map in a single threaded implementation with an Intel i5 processor at 1.6GHz (calculated over a set of 64 images of size 960×720). While the time complexity of approach is linear, the number of operations per pixel is much larger than other methods like LOCKY (chapter 4) or MSER. This is compensated by the fact that this technique acts as multiple feature detectors combined in a single framework, whilst other detectors find a single class of features.

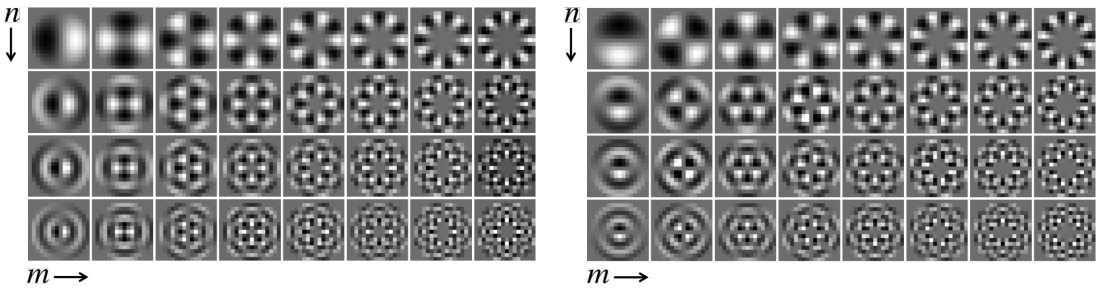


Figure 5.13: The 16×16 Bessel-Fourier kernels used for the detection of features. Real part of the kernels shown on the left, imaginary part on the right

5.5 Results

The repeatability measure introduced by Mikolajczyk *et al.* measures consistency of feature detectors across image deformations [75]. The test consists in detecting features in both the original image and a transformed version, then the detected features are mapped to the original space using a homography matrix to measure how well they overlap. In our comparison, features with 40% error or less are considered. We use the bikes, boat, graffiti, trees and ubc sequences from the Oxford affine-covariant regions dataset [75]; these sequences contain six images each with increasing blurring, perspective, jpeg compression, scale and rotation transformations. We also use the iguazu sequence introduced by Alcantarilla *et al.* [4], which presents increasing noise across images; and the semper and rome sequences presenting pure rotation transformations [39]. Figure 5.14 shows images from some of these sequences.

We compare our approach against various feature detectors available to the community, KAZE [4], SURF [12], LOCKY-S [62], BRISK [53], CenSurE [2] (marked as 'STAR'),

Figure 5.14: The first and sixth images of the boat, graffiti and semper sequences.

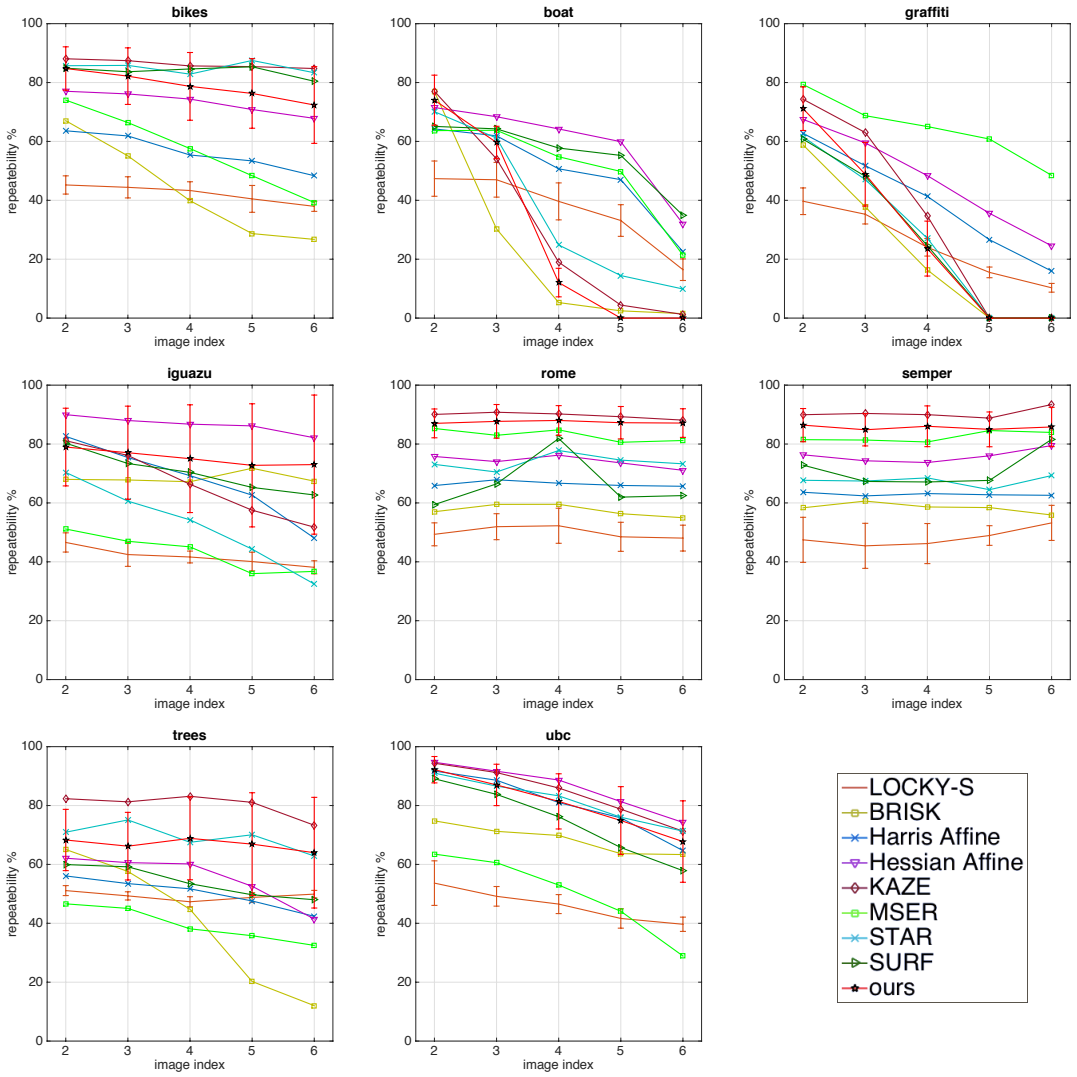


Figure 5.15: Results of the repeatability test presented in [75]. Our approach (marked as 'ours') is compared against against benchmark and state-of-the-art detectors. On average, the detector presented here is among the best performers in such test. The error bars signify the standard deviation across the repeatability of all 32 feature classes. Scale transformations seem to impose a more complex challenge as they change the intrinsic structure of features, on all other image deformations our approach performs consistently well.

Harris and Hessian affine [74] and MSER [72]. For KAZE[1] and LOCKY-S[2] we use the code provided by the authors. For SURF and BRISK we use the implementations provided in MATLAB, and for CenSurE the OpenCV code. CenSurE is known as the STAR detector in OpenCV, from hereafter we will use the latter name. KAZE and SURF both use four octaves with four intra-octave sub-levels, BRISK uses four octaves, STAR uses the default parameters provided in the library, LOCKY-S uses the parameters as presented in [62], our approach[3] detects features from 32 feature-maps at six scales

[1] https://github.com/pablofdezalc/kaze (ver. 1.8.0)
[2] https://github.com/jimmylomro/locky
[3] Our code is publicly available at https://github.com/jimmylomro/own

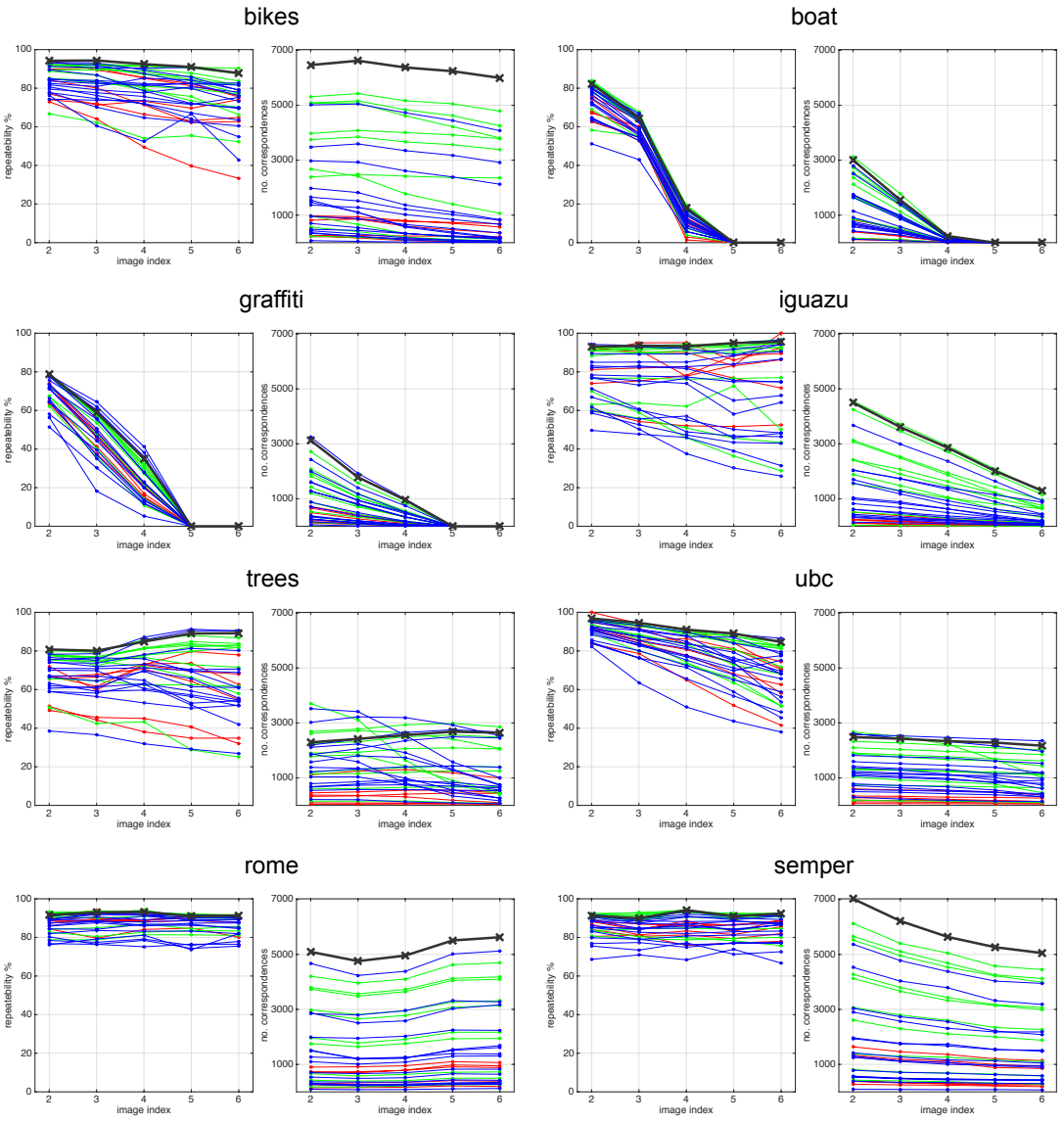


Figure 5.16: Every feature class is measured for repeatability. The results for repeatability (left on each group) and, the number of correspondences found in every feature class are shown (right on each group). Some features seem to perform better than others, the best average performer is highlighted with a thicker plot. The different symmetry groups are shown in different colours, non-symmetric features in blue, symmetric features in red and composite-symmetric features in green.

spanning 1.5 octaves always using 16×16 BF kernels (shown in figure 5.13). The results of our technique show the mean repeatability and standard deviation of all 32 feature classes, these results are shown in figure 5.15. These settings generate approximately 1500 to 2000 features per image, our algorithm detects the same amount on average over the 32 feature classes.

On average, our algorithm compares very well with the other detectors. Some features tend to perform better than others, the repeatability results of each of the 32 feature classes are shown in figure 5.16; the best average performer is highlighted with a thicker line this corresponds to the feature class E4 (figure 5.10). The main advantage of our

approach is that the user can choose the number of feature classes to use, allowing a denser sampling of the image. This advantage is well reflected by the number of correspondences (number of preserved features) across image transformations, providing an idea on the amount of features that every map provides. The number of correspondences found for every feature class is also shown in figure 5.16. The total number of detected features across all 32 classes in a 1024×680 image is on average $62,430$.

Figure 5.16 portrays the difference in performance for the different symmetry groups. Shown in blue are the non-symmetric features; using the coordinates of figure 5.10, these are A1, E1, H1, A2, B2, F2, H2, A3, D3, F3, H3, D4, G4 and H4. The symmetric features are shown in red, these are B1, C1, F1, E2, C2 and E3. And in green the composite-symmetric features, these are D1, D2 G2, B3, C3, G3, A4, B4, C4, E4 and F4. The composite-symmetric group tends to produce many more features than the other two groups, and their repeatability is usually higher. The symmetric features produce the fewest number of features whilst maintaining high repeatability scores. This may be due to there being many more step-edge pixels than roof-edge pixels.

5.6 Conclusion

We have presented a method that uses unsupervised learning to perform rotation-invariant detection structures, allowing the use of machine learning for generic feature detection. Our approach learns filters that show strong response on edge-like structures, a property that has been highlighted by representation learning algorithms. Showing very competitive repeatability scores on average, the use of any of the feature classes is a good performing detector. Moreover, our method allows the user to to choose the number of model structures thus, the user can substantially increase the amount of detected features in a controlled manner. This property could be exploited by tracking, texture recognition, image retrieval or other feature detection dependent algorithms.

Further research could be developed on a neural network based on our approach. We have shown how to train a single unit to create feature-maps from a rotation-invariant space, many of these units could be stacked in a multilayer architecture to build a rotation-invariant deep predictor.

The only recognised exception to the symmetry groups are $S_{6,1}$ and $S_{2,1}$. In theory these structures belong to the symmetric group, nonetheless as $k = 1$, all BF moments may have non-zero magnitude. The difference between these and S_0 lies on phase-congruency. A fourth symmetry group can be identified whose members are S_5 and S_{const}, we label it as the infinite-symmetric group. The analysis of these particular cases is still open for research. Furthermore, the phase of the BF moments is shown to contain crucial information. A method to consider such information could yield a more consistent detector as noisy features could be discriminated.

Chapter 6

Symmetry detection results

The implementation of symmetry detection using the feature detectors presented in chapters 4 and 5, is shown here. Discussion on the advantages and disadvantages of both approaches is also presented in this chapter.

The true positive rate and the number of false positives are calculated to compare our approach to [66] and [52]. Detected axes of symmetry are compared to ground-truth annotated images. Using LOCKY as the feature detection strategy yields the shortest computation times. Results obtained using the detection of learnt structures are consistently coherent with human annotations.

6.1 LOCKY

Figure 6.1 shows the implementation of LOCKY as the feature detection strategy for symmetry detection. The parameters used are the same as LOCKY-1, *i.e.* 10^5 iterations with size range 2^3 to 2^5 and a 12% threshold. The matching uses a minimum hamming distance of 120 and, DBSCAN uses $\epsilon = 0.2^2$ and $MinPts = 10$. Results of such experiment are shown in figure 6.2. Axes are successfully found on images with large or clearly evident symmetric structures.

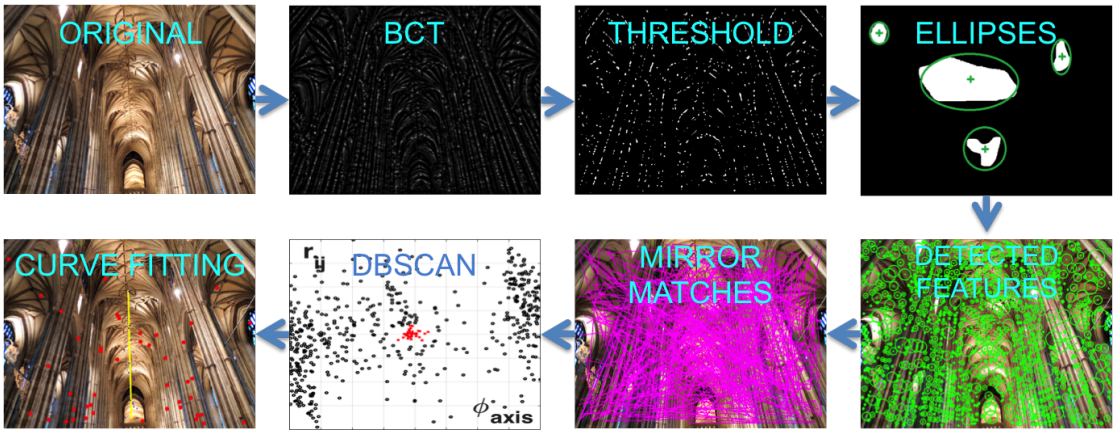


Figure 6.1: The implementation of LOCKY for symmetry detection. Features have good dispersion, covering the whole extent of images.

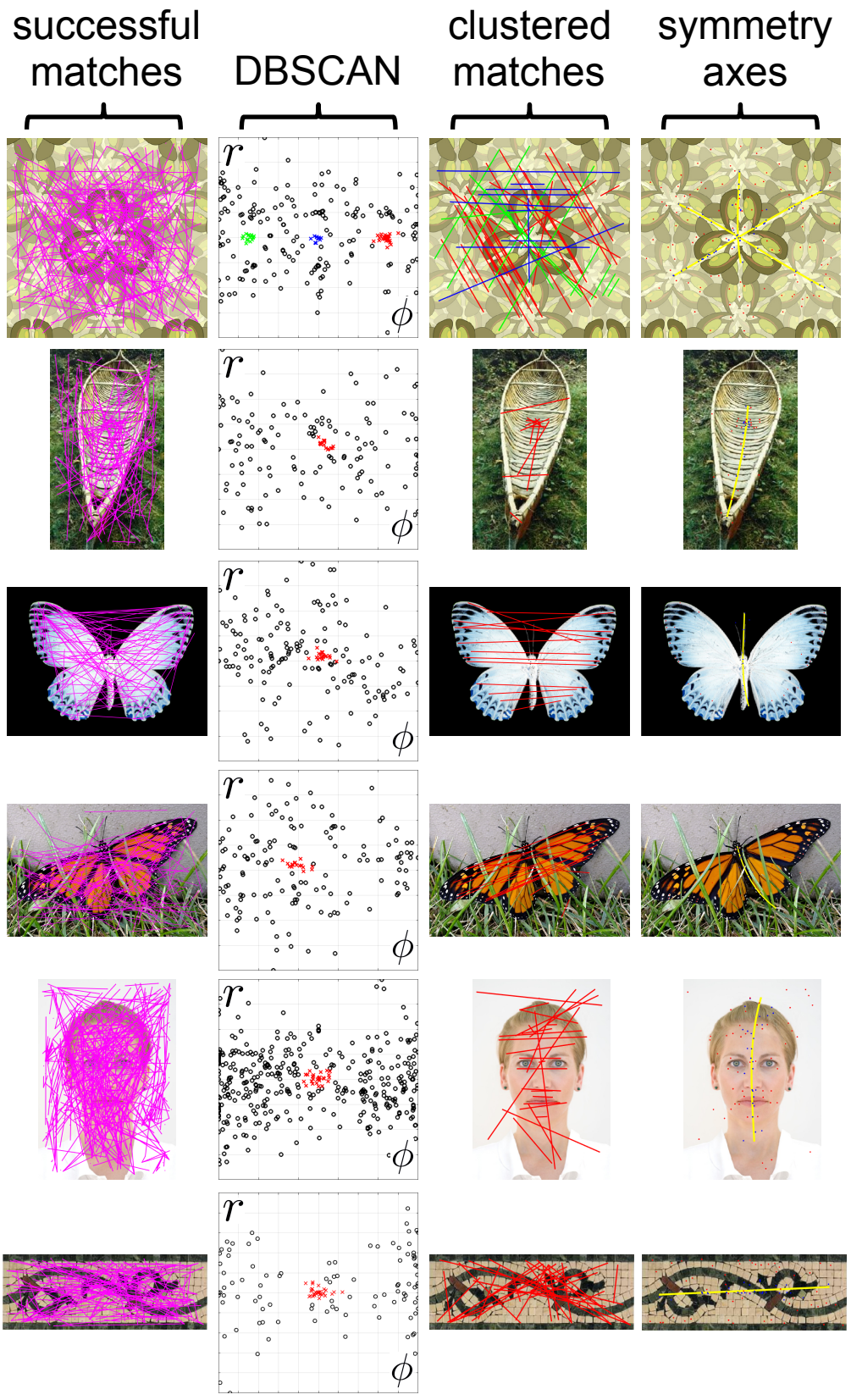


Figure 6.2: Symmetry results using LOCKY. All successful matches (left) are mapped into the parameter-space (mid-left). LOCKY produces many outlier matches, the clustering strategy in the parameter-space is robust enough to detect the correct axes.

Figure 6.3 shows two representative fail cases. Symmetric structures are smaller in the top row, the lack of symmetric matches supporting the axes causes the algorithm to fail. The axes to detect are many in the bottom row, again the parameter-space is poorly sampled and not all axes are detected. Intuitively, the detection could be improved in both scenarios using either of the following strategies:

- Accepting weaker matches. This increases the number of points in the parameter space, but it also introduces many more outliers. These outliers could cause false-positive detections or curve the detected axes.
- Increase the number of detected features. In a general sense, lowering the threshold of a feature detector results in noisy detections with poor localisation or other artefacts. Also, the matching process has $O(P^2)$ complexity where P is the number of detected features (see figure 6.7). This could potentially make the matching the slowest process of symmetry detection.

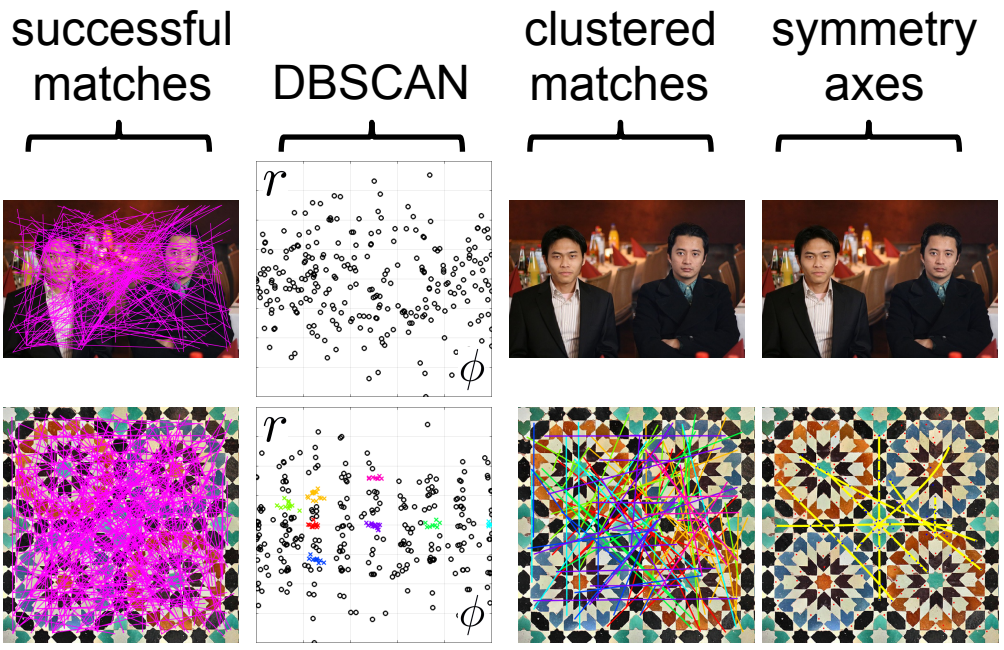


Figure 6.3: Symmetry detection fail cases using LOCKY. Smaller structures are more difficult to recognise. Also, images with many axes of symmetry need many more matching pairs to be detected. In both cases, the detection of symmetry would benefit from larger amounts of detected features.

6.2 Unsupervised learning to find structures in a rotation-invariant space

The implementation using the learnt structures as the feature detection technique is shown in figure 6.4.

It is possible to reduce one degree of freedom if the scale of all detected features is the same. Features can be detected at a single scale or, the scale of all detections at different scales can be fixed. The earlier approach yields a smaller number of detected features, while the latter eliminates distinctiveness amongst features, potentially causing noisy matches.

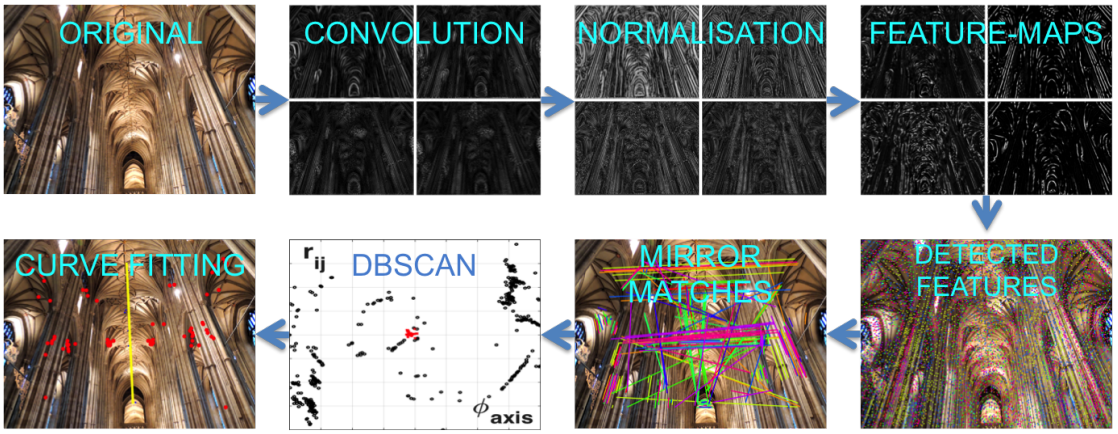


Figure 6.4: Use of the learnt structures for symmetry detection. Abundant sets of features provide a better sampling of the parameter-space.

The unsupervised learning for the rotation-invariant detection of structures eliminates the problem of small numbers of features, as each feature-map is a different detector. This allows the detection of features at a single scale. The image is rescaled so that $max(width, height) = 500$ pixels and, a kernel size of 16×16 for the BF-moment filters is used. This generates between 500 and 1000 features per feature-map. Other parameters for this implementation are: $M = 8$, $N = 1$, $C = 10$, $E_{thresh} = 5$ and $\tau = 0.6$.

To find symmetric pairs, only features of the same class are compared. Consider the total number of detected features P as a sum of the number of features detected in each class P_c. The number of descriptor comparisons when only matching features of the same class is $P_0^2 + P_1^2 + \cdots + P_{C-1}^2$. Whilst still quadratic, the number of operations performed is smaller than P^2. This reduces the computation time of the matching process.

Traditional feature detectors produce a set of points that belong to the same type of structure *i.e.* the same feature class. Several strategies can be implemented to partition such set into smaller subsets of points, for instance, the DoG detector could partition detections depending on the sign of the filtering response. Using the learnt structures, it is possible to partition the set into C feature classes, where C is a user defined parameter.

The complexity of a brute-force matching is $O(P^2)$ where P is the number of features. The partition $P = \sum_c P_c$ makes the complexity of the brute-force matching $O(\sum_c P_c^2)$. By the law of cosines $d^2 = a^2 + b^2 - 2abcos(\hat{ab})$ and its extension to polygons [23], the inequality 6.1 is always true.

$$P^2 > \sum_c P_c^2. \tag{6.1}$$

In the best case scenario $P_c = 1 \; \forall \; c$, where there are as many feature classes as there are features, the complexity is $O(C)$. On the other hand, when all features belong to the same class the complexity is $O(P^2)$. Therefore, the complexity of the brute-force matching remains $O(P^2)$. Instead of partitioning the set of detected features, we consider each class as a separate detector. For instance, if few features are needed, only one feature map is computed; and to extract more features we increment the number of feature classes used, whilst the number of detected features per class P_c remains constant for each class. Thus, the number of features P_c equals some constant, which is a linear factor of the best case scenario, in which the time complexity is $O(C)$, and C is the number of feature classes used. In this manner, the use of the learnt structures for detection reduces the complexity for the matching stage with a more flexible way of increasing the total number of detected features. This comes at the cost of a more time consuming detection stage.

Some of the results obtained using the learnt structures as the feature detection stage are presented in figure 6.5. Symmetric structures that were not detected using LOCKY are now detected. The parameter space shows clearer dense clouds of points, this is result of the detection and matching strategies. The better sampling of the image increases the probabilities of the axes to be detected, whilst matching only features of the same class maintains reduced computation times.

Two fail cases are shown in figure 6.6. In the bottom, there is a very small number of symmetric matches. The complex random pattern (dry grass) causes the descriptors to be much different from each other thus, the majority of them are considered unsuccessful matches. On the top, detection fails because of two different reasons: the intricate patterns is different on every petal, and the background has simpler structures. The combination of these conditions creates clusters of outliers that have as many symmetric matches as the actual axes of symmetry, generating both false-positive and false-negative detections.

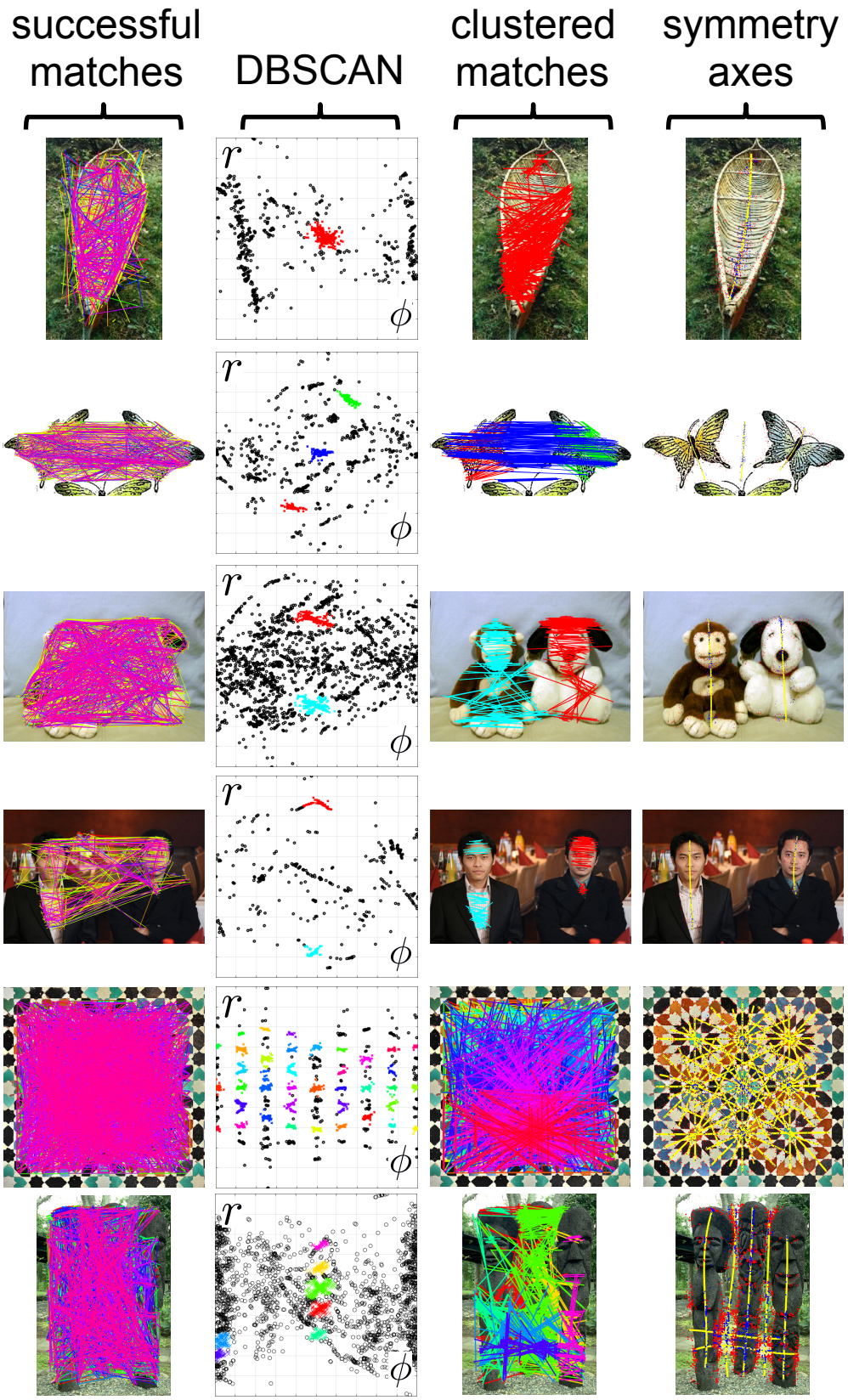


Figure 6.5: The abundance of features enhances the detection of smaller symmetrical structures, even with multiple axes. The number of detected features is large even in low textured images (second row from the top).

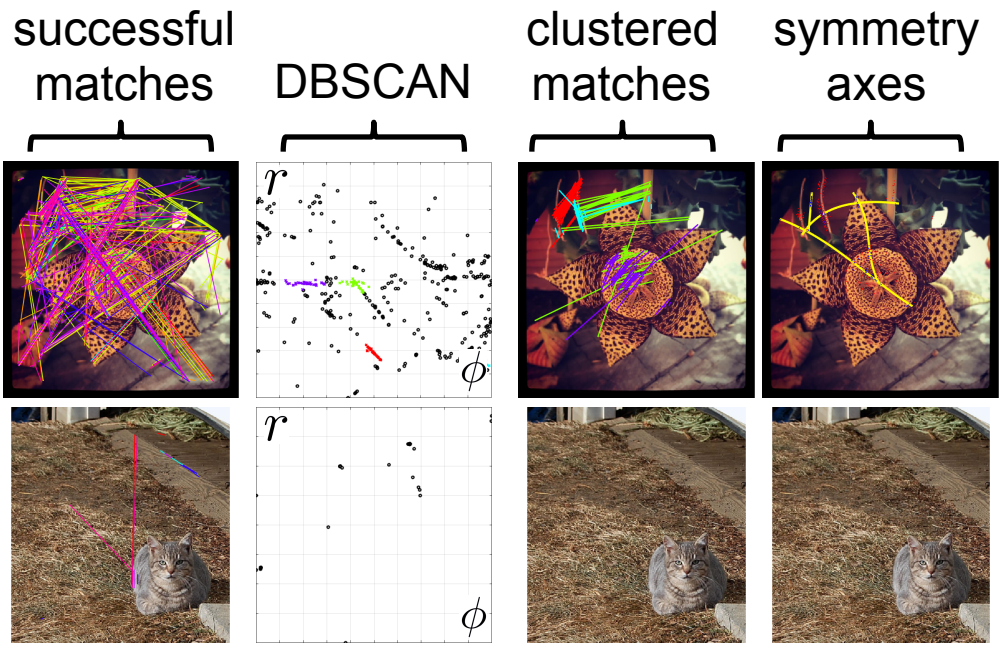


Figure 6.6: Symmetry detection fail cases using the learnt structures. Highly textured regions with no apparent order disrupt the detection.

6.3 Comparison

For a more comprehensive comparison, timings of the new method are compared with other approaches. Having a considerably larger amount of features, the computation time is expected to grow. The rotation-invariant detection of features from the learnt structures is in general a slow process. Table 6.1 shows the average detection times calculated on a set 127 images rescaled with $max(width, height) = 500$. To benchmark the detection of features we also use the Difference of Gaussians (DoG) detection strategy of SIFT [65]. For this experiment, the settings used for LOCKY and the learnt structures those shown in sections 6.1 and 6.2 respectively. The parameters used by the DoG are the ones in the original implementation.

Method	**Timing (ms)**
DoG	105
LOCKY-1	21
Learnt structures	194 (19.4/map)

Table 6.1: Feature detection times on rescaled images.

Figure 6.7 demonstrates that scaling the number of features using multiple features classes, is better than the use of a single feature class. The total number of detected features is plotted against the computation time of the matching stage. Moreover, the use of binary descriptors also reduces the computation time. Times plotted in blue use 128 dimensions floating point descriptors for each feature *i.e.* SIFT description, red plots are the binary descriptors *i.e.* BRISK. Dashed lines indicate all features are compared

to each other (single feature class), continuous lines indicate only features of the same class are compared against each other (multiple feature classes).

In this experiment we extracted features from a total of 64 images using the learnt structures, and use BRISK and SIFT to extract descriptors from the detections. For the single feature class we use the accumulated of all features and match all-to-all, for the multiple feature classes we match only features of the same class. The values on the x axis are generated with the average number of features generated across all 64 images, *i.e.* for the dashed lines this is P and for the solid lines this is $\sum_c P_c$ (each circle marker is an increment of c, section 6.2).

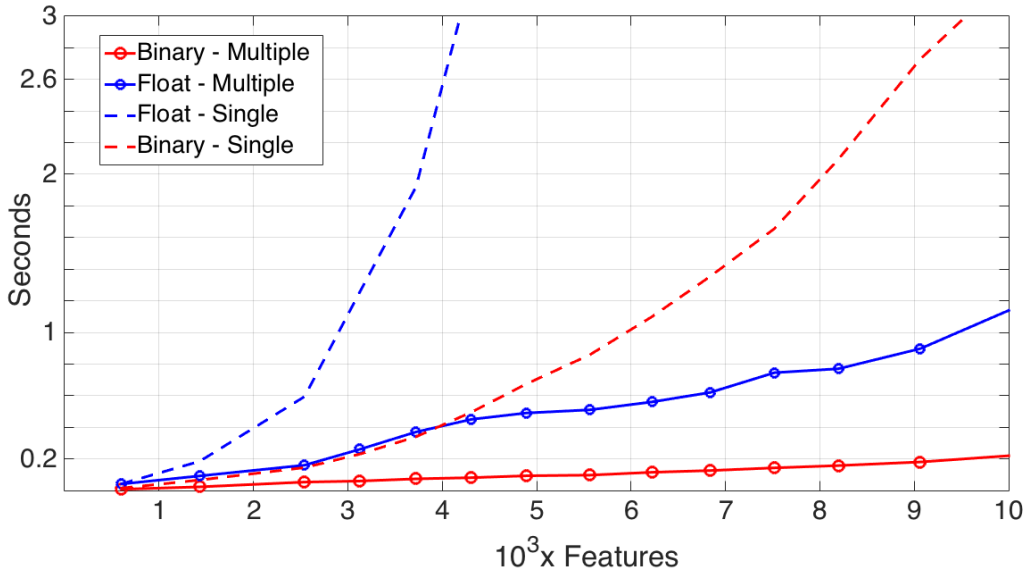


Figure 6.7: Comparison between the matching time of traditional descriptors (blue) and binary descriptors (red). Timings using a single feature class (match all-to-all) and multiple feature classes (match only features of the same class) are shown. The matching stage becomes tractable when only features of the same class are compared against each other.

For a quantitative comparison, we use the true positive rate of the detections on the 64 images of the symmetry detection PAMI2012 dataset [52]. The true positive rate is defined as the number of detected axes divided by the total number of human-annotated ground-truth axes. To obtain such results, the output axes of the algorithms are visually compared against the ground-truth. If a detected axis captures a symmetric object that is annotated in the ground-truth, the axis is marked as a true-positive. We also use the number of false-positives, these are the detections by the algorithms that do not have a corresponding ground-truth annotation. Results of this experiment are shown in table 6.2. Results using DoG, LOCKY and learnt structures for feature detection[1] are shown.

The settings for LOCKY and the learnt structures used to obtain the results in table 6.2 are those shown in sections 6.1 and 6.2 respectively. The settings used for the DoG

[1]Code available at https://github.com/jimmylomro/symmetry

detector are those presented by Lowe in the original implementation [65]. Our approach compares to the state-of-the-art using only 6% of the computation time. We provide result images commonly used in the symmetry detection literature below, this is to give the reader a means to visually compare our results with those obtained by other authors. Figure 6.8 shows our results alongside ground truth annotations from [59]. Figure 6.9 compares our results with those obtained in [59].

Method	**Symmetry** True Positive Rate (# False Positives)				**Timings**
	Straight	Straight Glide	Curved	Curved Glide	mean(std) secs
proposed (DoG)	81%(6)	40%(7)	62%(5)	40%(6)	0.332(0.06)
proposed (LOCKY)	71%(8)	33%(9)	50%(7)	40%(7)	0.094(0.03)
proposed (learnt)	91%(4)	80%(6)	83%(6)	70%(7)	0.59(0.26)
Lee [52]	86%(3)	80%(3)	83%(3)	60%(4)	9.7(10.3)
Loy [66]	91%(9)	7%(46)	0%(38)	0%(26)	6.1(9.4)

Table 6.2: Results comparing our approach to those showed by Lee and Liu in [52].

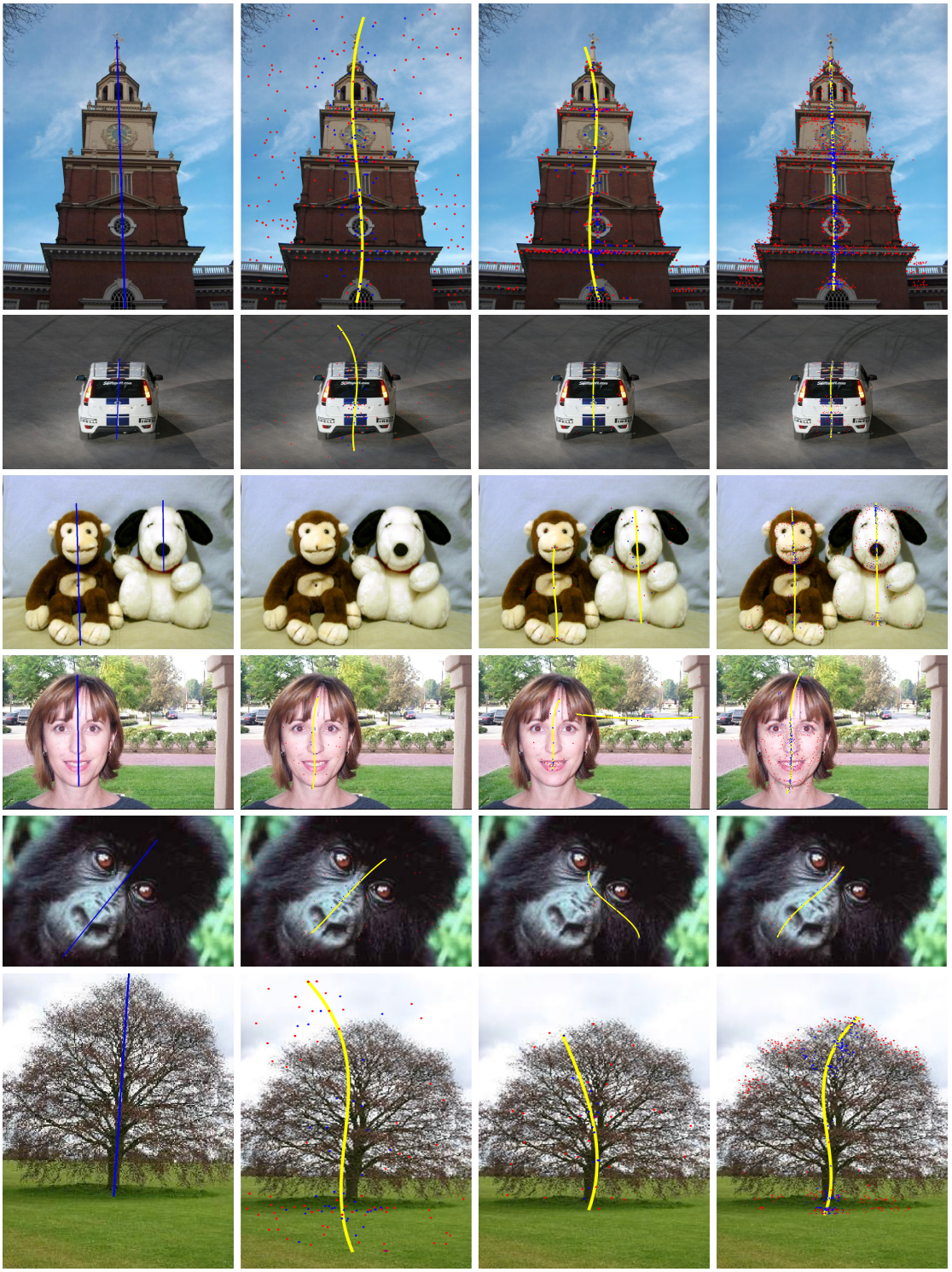


Figure 6.8: Human annotated ground truth [59] (left), compared to the symmetry detection results using LOCKY, DoG and, learnt structures as the feature detection strategies (left to right). The axes of symmetry are better sampled (blue dots) with the learnt structures method, such detections are closer to the ground truth.

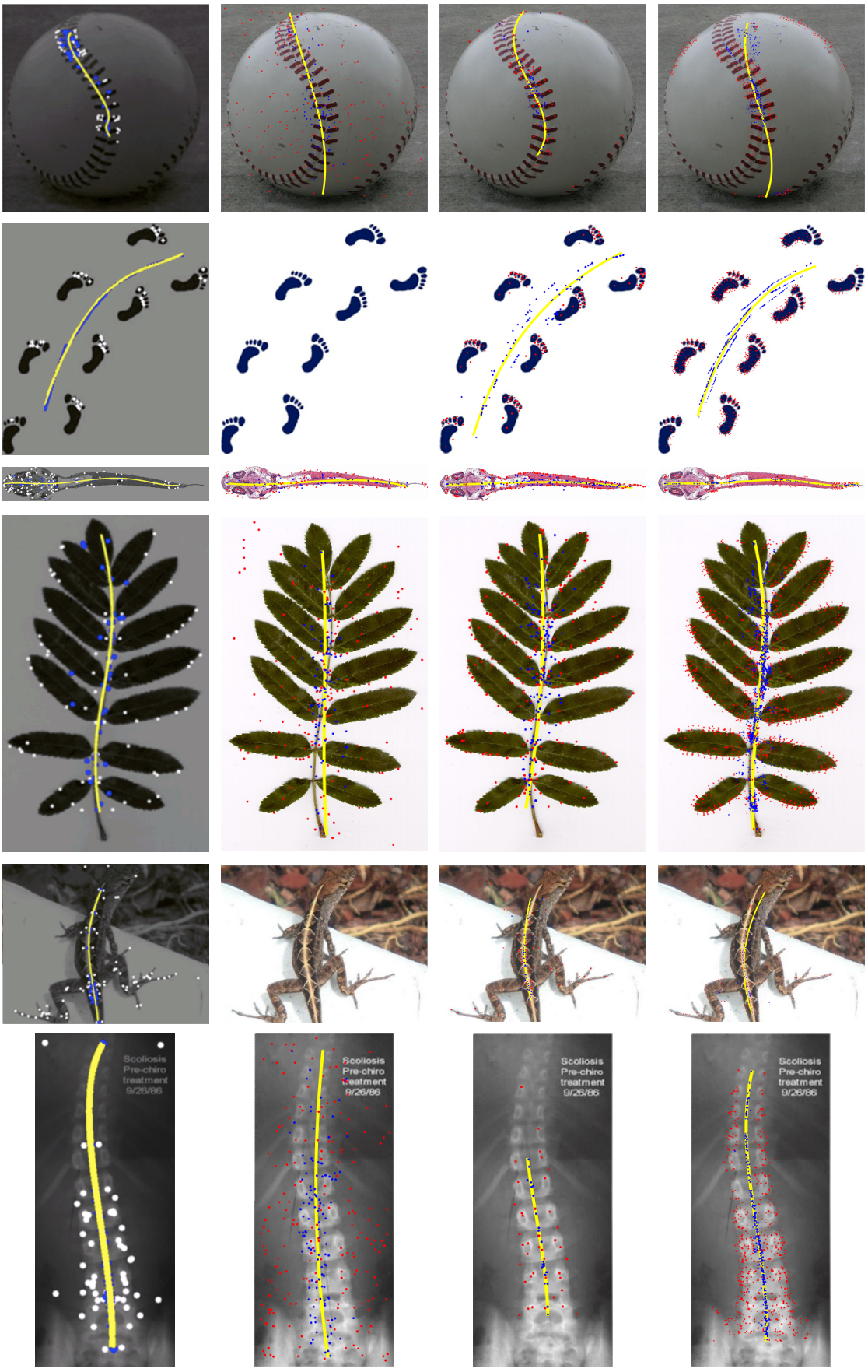


Figure 6.9: Results obtained by Lee [52] (left), compared against detections with the proposed method. Results obtained using LOCKY, DoG and learnt structures shown from left to right.

6.4 Results

The results in this section were obtained using the learnt structures as the feature detection stage.

Straight and curved reflection symmetries with single and multiple axes are shown in figure 6.10. Red dots are the features supporting the axes, blue dots are the middle points of the matching pairs. The large amounts of detected features aid the detection of multiple axes in an image, *e.g.* the starfish. Also, plentiful sets of features are detected even in low textured images, *e.g.* the footprints image.

Increasing the matching threshold $BRISK_{thresh}$ generates more outliers in the parameter space, at the same time, it increases the probability of detecting cluttered axes. Robustness of DBSCAN for outliers lessens these negative effects. This is the case of the lizard and the caterpillar images.

Figure 6.10: Straight and curved reflection axes detection using the learnt structures as the detection strategy.

Reflection and glide-reflection symmetries are usually not mutually exclusive. For instance, repetitive patterns with a translation generate both reflection and glide-reflection symmetry at the same time *e.g.* the leafs image in figure 6.10. Misalignment along the axis differentiates these cases, some examples are present in figure 6.11.

Repetitive patterns that show glide-reflection symmetry encompass multiple axes *i.e.* the translation is periodic. In straight axes, this is not evident as all axes are aligned *e.g.* the beads necklace image. On the other hand curved glide-symmetries deviate such axes, this can be seen in the image of footprints in figure 6.11. Misaligned lines of blue dots voting for different axes are recognisable. In extreme cases, this can cause false-positive detections.

Figure 6.11: Straight and curved glide-reflection axes detection using the learnt structures as the detection strategy.

Wallpaper patterns present several axes of symmetry and are usually low-textured, yielding a more complex task (figure 6.12). The abundance of features is important to densely sample the parameter space, our technique is particularly good in such images. With thousands of detected features, the probability of matching pairs to exist is larger thus, all axes are densely sampled and detected.

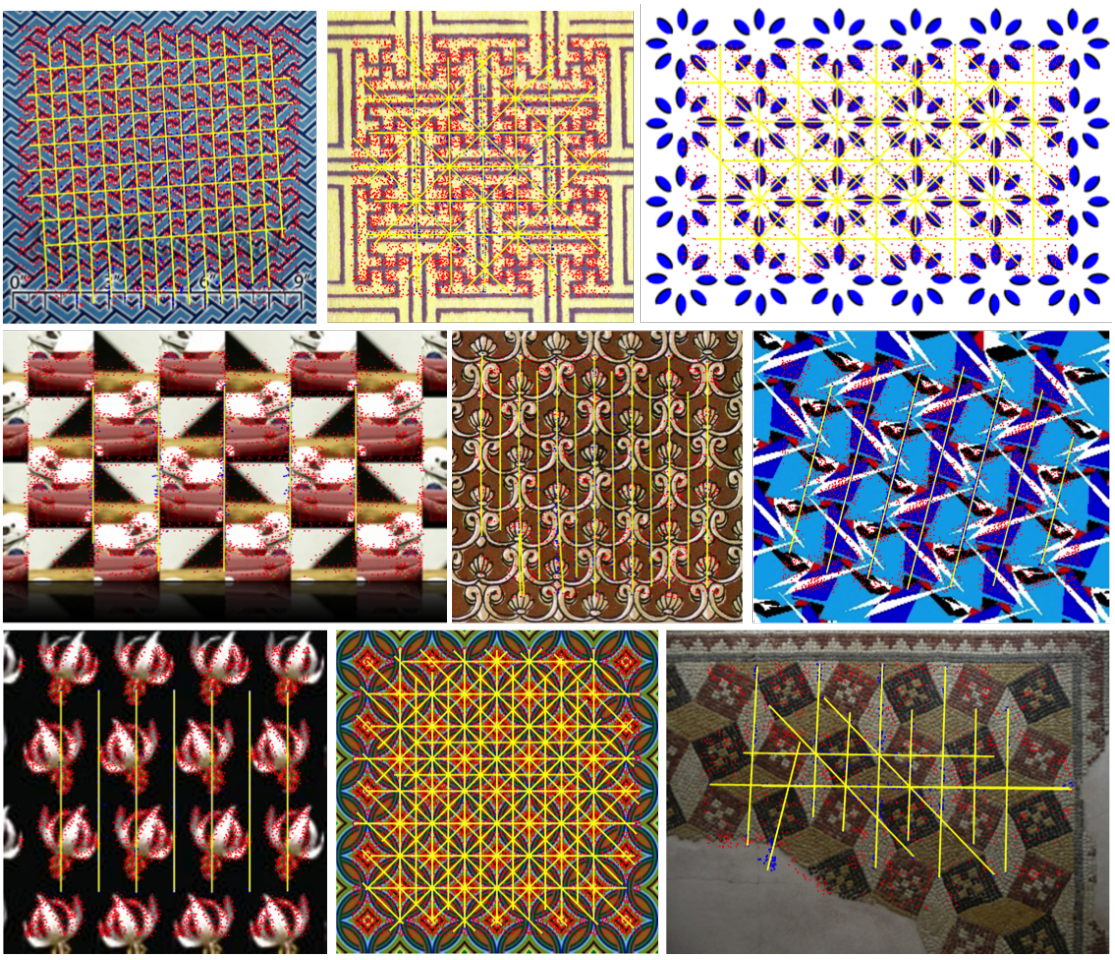


Figure 6.12: Detection results on wallpaper patterns using the learnt structures as the detection strategy.

Figure 6.13 shows four representative fail cases. The spine and the snake show very abrupt deformations, in such images it is difficult to outline the stage at which the algorithm fails. Accepting weaker matches could be beneficial to overcome image deformations, although too many outliers could cause false-positive detections as is the case in the left-most image. The smaller axis on the top-left corner of the middle-right image, is in fact a true-positive that could be easily overlooked by humans. Simple structures that create such symmetries are often the case of failure, as these create many outliers in the parameter space.

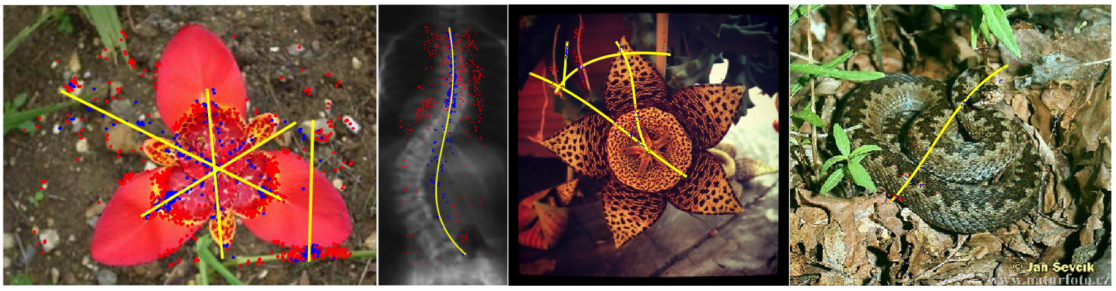


Figure 6.13: Miscellaneous fail cases using the learnt structures as the detection strategy.

Chapter 7

Conclusions and future work

7.1 Conclusions

The distinction between cognition levels of symmetry shown in chapter 1, improves the understanding of the different approaches for the automated detection of symmetry.

The feature based methods provide a good alternative to deal with image transformations and artefacts. Such approaches are usually computationally expensive, making them less attractive for implementation. The work shown in this thesis, proves that symmetry detection can be a rapid task.

Feature detection allows the reduction of a search space, where features are representative areas in the image. Feature based symmetry detection takes advantage of this to find similar regions in images. In chapter 3, three major speed issues with feature-based methods were highlighted: feature description, descriptor mirroring and parameter-space analysis.

The use of binary descriptors for symmetry, largely improves the speed of the process (figure 6.7). Additionally, only comparing features of the same class drastically reduces the computation time of the matching stage. Overall the time complexity of the symmetry detection algorithm is linear on the number of pixels in the image, but quadratic on the number of successful matches. This is due to the complexity of DBSCAN. The analysis of the parameter-space presented in this thesis avoids its discretisation, allowing the detection of curved axes of reflection and glide-reflection symmetries. Percentages in table 6.2, reflect the advantage of using such approach.

Chapters 4 and 5, show two new proposals for feature detection. LOCKY is one of the fastest affine-covariant blob detectors (table 4.1). The non-deterministic nature of the BCT impacts the repeatability results in a negative manner, nonetheless, LOCKY shows competitive scores compared with the state of the art (figure 4.9). Although symmetry detection is fastest when LOCKY is used as the feature detection stage (section 6.1), the relatively limited amounts of features impairs the detection, reducing the number of true-positives.

Using unsupervised learning to perform rotation-invariant detection of structures, enables the search for inherently repetitive patterns. This is evidenced by the repeatability results

shown in figure 5.15. This new feature detection method also introduces an alternative to the manual selection of model structures. In this manner, it is possible to substantially increase the number of detected features (figure 5.16). The large amounts of features benefit the detection of symmetry obtaining state of the art results compared to other approaches (table 6.2).

Implementing these enhancements to the feature based approach, a large improvement in computation time is achieved. Yet many issues remain to be solved in the area of automated symmetry detection. Feature based symmetry detection has multiple stages, when one of these stages fails the entire process fails. The images in figure 6.13 portray such cases. The following section describes possible alternatives and research directions relevant to the symmetry and feature detection areas.

Visual cues are formed by the complex interactions of multiple forming waves, we have explained this by means of the Fourier transform in cylindrical coordinates in chapter 5. The classification of low-level primitives into non-symmetric, symmetric and composite-symmetric groups is a step forward on the understanding of structure.

7.2 Future work

As low-levels of symmetry are usually related to the detection of features, the interactions between composing waves should be researched in more depth. Symmetric structures arise when there is phase congruency, but the lack of it does not diminish the presence of characteristic structures. Feature and symmetry detection could benefit from the use of phase information to find more specific patterns.

Our strategy for feature based symmetry detection makes use of the line parameters ϕ_{axis} and r_{ij} (chapter 3). Curved axes of symmetry may be better described by a different set of parameters. The technique could also be used for rotational symmetry [66], the parameter space for this application should also be investigated.

The use of neural networks for skeletonization highlights the potential of this approach for the detection of symmetry [32, 93]. These methods extract low levels of symmetry, their use for the detection of mid levels of symmetry remains unresearched. Also, these algorithms rely on human-annotated data which is highly appreciation and application dependent.

The method presented in chapter 5 can be extended in several aspects. Our approach trains models from data directly extracted from images, stacking these units in a multilayer architecture could yield a rotation-invariant deep predictor. Rotation-invariant CNN frameworks have been proposed before and can be thoroughly extended [69, 27].

Our technique can be parallelised to run in a graphics processing unit, this could drastically increase the speed of the detection. The computation time of the symmetry detection also improves with a faster feature detection. A real-time detection of symmetry could unlock its application in different areas.

The calculation of E_{thresh} in chapter 5 could be simplified if the gain of the filters (the BF kernels) is normalised. Further analysis is needed to understand how incrementing the variables M and N affect the calculation of signal energy in such range.

Another possible avenue is to investigate the intrinsic relation between feature detection and description. Detection of features aims to find general kinds of structures that are similar to each other, whilst descriptors are ideally unique labels. Bessel-Fourier moments are a complete and orthogonal set of basis functions, this means that a large-enough set of moments can uniquely describe an image-patch. This thesis has shown that clustering the magnitude of a few BF moments creates generic models of similar structures. A possible reason for this might be that low-frequency components are general descriptors whilst higher frequency components are more specific.

Closing the gap between feature detection and description could lead to a more unified symmetry operator. As explained in section 5.3.1, every pixel in the image is mapped into an $M \times N$-dimensional rotation-invariant space. If this representation is specific enough to uniquely describe patterns, the symmetry detection could be directly performed at this stage, obviating the need for feature description and matching. In this manner, the operator would use all pixels in the image without the need to reduce the search space with feature detection.

References

[1] Radhakrishna Achanta, Appu Shaji, Kevin Smith, Aurelien Lucchi, Pascal Fua, and Sabine Süsstrunk. Slic superpixels compared to state-of-the-art superpixel methods. *Transactions on Pattern Analysis and Machine Intelligence*, 34(11):2274–2282, 2012.

[2] Motilal Agrawal, Kurt Konolige, and Morten Rufus Blas. Censure: Center surround extremas for realtime feature detection and matching. In *European Conference on Computer Vision*, pages 102–115. Springer, 2008.

[3] Alexandre Alahi, Raphael Ortiz, and Pierre Vandergheynst. Freak: Fast retina keypoint. In *Conference on Computer Vision and Pattern Recognition*, pages 510–517. IEEE, 2012.

[4] Pablo Fernández Alcantarilla, Adrien Bartoli, and Andrew J Davison. Kaze features. In *European Conference on Computer Vision*, pages 214–227. Springer, 2012.

[5] Pablo Fernández Alcantarilla, Jesus Nuevo, and Adrien Bartoli. Fast explicit diffusion for accelerated features in nonlinear scale spaces. *Transactions on Pattern Analysis and Machine Intelligence*, 34(7):1281–1298, 2011.

[6] Ibragim Atadjanov and Seungkyu Lee. Bilateral symmetry detection based on scale invariant structure feature. In *International Conference on Image Processing*, pages 3447–3451. IEEE, 2015.

[7] Artem Babenko, Anton Slesarev, Alex andr Chigorin, and Victor Lempitsky. Neural codes for image retrieval. In *European Conference on Computer Vision*, pages 584–599. Springer, 2014.

[8] Li Bai, Linlin Shen, and Yan Wang. A novel eye location algorithm based on radial symmetry transform. In *International Conference on Pattern Recognition*, volume 3, pages 511–514. IEEE, 2006.

[9] Vassileios Balntas, Lilian Tang, and Krystian Mikolajczyk. Bold-binary online learned descriptor for efficient image matching. In *Conference on Computer Vision and Pattern Recognition*, pages 2367–2375, 2015.

[10] Luca Baroffio, Alessandro EC Redondi, Marco Tagliasacchi, and Stefano Tubaro. A survey on compact features for visual content analysis. *Transactions on Signal and Information Processing*, 5:e13, 2016.

[11] Harry Bateman. *Higher Transcendental Functions, Volume II*. McGraw-Hill, 1953.

[12] Herbert Bay, Andreas Ess, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded-up robust features. *Computer Vision and Image Understanding*, 110(3):346–359, 2008.

[13] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European Conference on Computer Vision*, pages 404–417. Springer, 2006.

[14] Saeid O Belkasim, Malayappan Shridhar, and Majid Ahmadi. Pattern recognition with moment invariants: a comparative study and new results. *Pattern Recognition*, 24(12):1117–1138, 1991.

[15] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[16] Harry Blum. A transformation for extracting new descriptors of shape. Cambridge, MA: MIT Press, 1967.

[17] F Warren Burton, John G Kollias, and Nikitas A Alexandridis. An implementation of the exponential pyramid data structure with application to determination of symmetries in pictures. *Computer Vision, Graphics, and Image Processing*, 25(2):218–225, 1984.

[18] Michael Calonder, Vincent Lepetit, Christoph Strecha, and Pascal Fua. Brief: Binary robust independent elementary features. In *European Conference on Computer Vision*, pages 778–792. Springer, 2010.

[19] Michael Chertok and Yosi Keller. Spectral symmetry analysis. *Transactions on Pattern Analysis and Machine Intelligence*, 32(7):1227–1238, 2010.

[20] Mircea Cimpoi, Subhransu Maji, and Andrea Vedaldi. Deep filter banks for texture recognition and segmentation. In *Conference on Computer Vision and Pattern Recognition*, pages 3828–3836, 2015.

[21] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *International conference on artificial intelligence and statistics*, pages 215–223, 2011.

[22] Adam Coates and Andrew Y Ng. Learning feature representations with k-means. In *Neural Networks: Tricks of the Trade*, pages 561–580. Springer, 2012.

[23] Lee Collins and Thomas J Osler. Law of cosines generalised for any polygon and any polyhedron. *The Mathematical Gazette*, 95(533):240–243, 2011.

[24] Vito Di Gesu and Cesare Valenti. The discrete symmetry transform in computer vision. 1995.

[25] Sven J Dickinson, Alex Levinshtein, Pablo Sala, and Cristian Sminchisescu. The role of mid-level shape priors in perceptual grouping and image abstraction. In *Shape Perception in Human and Computer Vision*, pages 1–19. Springer, 2013.

[26] Sven J Dickinson and Zygmunt Pizlo. *Shape perception in human and computer vision*. Springer, 2013.

[27] Sander Dieleman, Kyle W Willett, and Joni Dambre. Rotation-invariant convolutional neural networks for galaxy morphology prediction. *Monthly notices of the royal astronomical society*, 450(2):1441–1459, 2015.

[28] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *International Conference on Knowledge Discovery and Data Mining*, volume 96, pages 226–231, 1996.

[29] Clement Farabet, Camille Couprie, Laurent Najman, and Yann LeCun. Learning hierarchical features for scene labeling. *Transactions on pattern analysis and machine intelligence*, 35(8):1915–1929, 2013.

[30] Li Fei-Fei, Rob Fergus, and Pietro Perona. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories. *Computer Vision and Image Understanding*, 106(1):59–70, 2007.

[31] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

[32] Christopher Funk and Yanxi Liu. Beyond planar symmetry: Modeling human perception of reflection and rotation symmetries in the wild. *arXiv preprint arXiv:1704.03568*, 2017.

[33] Steven Grant and Laurent Itti. Saliency mapping enhanced by symmetry from local phase. In *International Conference on Image Processing*, pages 653–656. IEEE, 2012.

[34] Lewis D Griffin. Symmetries of 2-d images: cases without periodic translations. *Journal of Mathematical Imaging and Vision*, 34(3):259–269, 2009.

[35] Lewis D Griffin and Martin Lillholm. Symmetry sensitivities of derivative-of-gaussian filters. *Transactions on Pattern Analysis and Machine Intelligence*, 32(6):1072–1083, 2010.

[36] Chris Harris and Mike Stephens. A combined corner and edge detector. In *Alvey Vision Conference*, volume 15, page 50. Manchester, UK, 1988.

[37] Daniel Cabrini Hauagge and Noah Snavely. Image matching using local symmetry features. In *Conference on Computer Vision and Pattern Recognition*, pages 206–213. IEEE, 2012.

[38] James B Hayfron-Acquah, Mark S Nixon, and John N Carter. Automatic gait recognition by symmetry analysis. *Pattern Recognition Letters*, 24(13):2175–2183, 2003.

[39] Jared Heinly, Enrique Dunn, and Jan-Michael Frahm. Comparative evaluation of binary features. In *European Conference on Computer Vision*, pages 759–773. Springer, 2012.

[40] Geoffrey E Hinton, Simon Osindero, and Yee-Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[41] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *Transactions on Information Theory*, 8(2):179–187, 1962.

[42] Herve Jegou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Perez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *Transactions on Pattern Analysis and Machine Intelligence*, 34(9):1704–1716, 2012.

[43] John Jonides, David E Irwin, and Steven Yantis. Integrating visual information from successive fixations. *Science*, 215(4529):192–194, 1982.

[44] Dong-Su Kim and Sung-Il Chien. Automatic car license plate extraction using modified generalized symmetry transform and image warping. In *International Symposium on Industrial Electronics*, volume 3, pages 2022–2027. IEEE, 2001.

[45] Gert Kootstra, Sjoerd De Jong, and Lambert RB Schomaker. Using local symmetry for landmark selection. In *Computer Vision Systems*, pages 94–103. Springer, 2009.

[46] Gert Kootstra, Arco Nederveen, and Bart De Boer. Paying attention to symmetry. In *British Machine Vision Conference*, pages 1115–1125. The British Machine Vision Association and Society for Pattern Recognition, 2008.

[47] Peter Kovesi. Symmetry and asymmetry from local phase. In *Australian Joint Conference on Artificial Intelligence*, volume 190. Citeseer, 1997.

[48] J Lopez Krahe. Detection of symmetric and radial structures in images. In *International Conference on Pattern Recognition*, pages 947–950, 1986.

[49] Andreas Kuehnle. Symmetry-based recognition of vehicle rears. *Pattern Recognition Letters*, 12(4):249–258, 1991.

[50] Brian Kulis and Michael I Jordan. Revisiting k-means: New algorithms via bayesian nonparametrics. In *International Conference on Machine Learning*, pages 513–520, 2012.

[51] Seungkyu Lee, Robert T Collins, and Yanxi Liu. Rotation symmetry group detection via frequency analysis of frieze-expansions. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[52] Seungkyu Lee and Yanxi Liu. Curved glide-reflection symmetry detection. *Transactions on Pattern Analysis and Machine Intelligence*, 34(2):266–278, 2012.

[53] Stefan Leutenegger, Margarita Chli, and Roland Y Siegwart. Brisk: Binary robust invariant scalable keypoints. In *International Conference on Computer Vision*, pages 2548–2555. IEEE, 2011.

[54] Alex Levinshtein, Cristian Sminchisescu, and Sven Dickinson. Multiscale symmetric part detection and grouping. *International Journal of Computer Vision*, 104(2):117–134, 2013.

[55] Cheng-Chung Lin and Wei-Chung Lin. Extracting facial features by an inhibitory mechanism based on gradient distributions. *Pattern Recognition*, 29(12):2079–2101, 1996.

[56] Tony Lindeberg. Edge detection and ridge detection with automatic scale selection. In *Conference on Computer Vision and Pattern Recognition*, pages 465–470. IEEE, 1996.

[57] Tony Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):79–116, 1998.

[58] Haihui Liu, Wan-Lei Zhao, Hanzi Wang, Kyungmo Koo, and Sangwhan Moon. A comparative study on features for similar image search. In *International Conference on Internet Multimedia Computing and Service*, pages 349–353. ACM, 2016.

[59] Jingchen Liu, George Slota, Gang Zheng, Zhaohui Wu, Minwoo Park, Seungkyu Lee, Ingmar Rauschert, and Yanxi Liu. Symmetry detection from realworld images competition 2013: Summary and results. In *Computer Vision and Pattern Recognition Workshops*, pages 200–205. IEEE, 2013.

[60] Li Liu, Yunli Long, Paul W Fieguth, Songyang Lao, and Guoying Zhao. Brint: binary rotation invariant and noise tolerant texture classification. *Transactions on Image Processing*, 23(7):3071–3084, 2014.

[61] Jaime Lomeli-R and Mark S Nixon. The brightness clustering transform and locally contrasting keypoints. In *International Conference on Computer Analysis of Images and Patterns*, pages 362–373. Springer, 2015.

[62] Jaime Lomeli-R and Mark S Nixon. An extension to the brightness clustering transform and locally contrasting keypoints. *Machine Vision and Applications*, 27(8):1187–1196, 2016.

[63] Jaime Lomeli-R and Mark S Nixon. Learning salient structures for the analysis of symmetric patterns. In *International Conference on Image Analysis and Recognition*, pages 286–295. Springer, 2017.

[64] Jonathan L Long, Ning Zhang, and Trevor Darrell. Do convnets learn correspondence? In *Advances in Neural Information Processing Systems*, pages 1601–1609, 2014.

[65] David G Lowe. Object recognition from local scale-invariant features. In *International Conference on Computer Vision*, volume 2, pages 1150–1157. IEEE, 1999.

[66] Gareth Loy and Jan-Olof Eklundh. Detecting symmetry and symmetric constellations of features. In *European Conference on Computer Vision*, pages 508–521. Springer, 2006.

[67] Gareth Loy and Alexander Zelinsky. Fast radial symmetry for detecting points of interest. *Transactions on Pattern Analysis and Machine Intelligence*, 25(8):959–973, 2003.

[68] Elmar Mair, Gregory D Hager, Darius Burschka, Michael Suppa, and Gerhard Hirzinger. Adaptive and generic corner detection based on the accelerated segment test. In *European Conference on Computer Vision*, pages 183–196. Springer, 2010.

[69] Diego Marcos, Michele Volpi, and Devis Tuia. Learning rotation invariant convolutional filters for texture classification. In *International Conference on Pattern Recognition*, pages 2012–2017. IEEE, 2016.

[70] David Marr and Ellen Hildreth. Theory of edge detection. *Royal Society of London B: Biological Sciences*, 207(1167):187–217, 1980.

[71] David Marr and A Vision. A computational investigation into the human representation and processing of visual information. *WH San Francisco: Freeman and Company*, 1982.

[72] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and Vision Computing*, 22(10):761–767, 2004.

[73] Max Mignotte. Symmetry detection based on multiscale pairwise texture boundary segment interactions. *Pattern Recognition Letters*, 74:53–60, 2016.

[74] Krystian Mikolajczyk and Cordelia Schmid. An affine invariant interest point detector. In *European Conference on Computer Vision*, pages 128–142. Springer, 2002.

[75] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc Van Gool. A comparison of affine region detectors. *International Journal of Computer Vision*, 65(1-2):43–72, 2005.

[76] R Mukundan, SH Ong, and Poh Aun Lee. Image analysis by tchebichef moments. *Transactions on Image Processing*, 10(9):1357–1364, 2001.

[77] Ramakrishnan Mukundan and KR Ramakrishnan. *Moment functions in image analysis: theory and applications*, volume 100. World Scientific, 1998.

[78] David Nistér and Henrik Stewénius. Linear time maximally stable extremal regions. *European Conference on Computer Vision*, pages 183–196, 2008.

[79] Viorica Patraucean, Rafael Grompone von Gioi, and Maks Ovsjanikov. Detection of mirror-symmetric image patches. In *Computer Vision and Pattern Recognition Workshops*, pages 211–216, 2013.

[80] Felipe Pegado, Kimihiro Nakamura, Laurent Cohen, and Stanislas Dehaene. Breaking the symmetry: mirror discrimination for single letters but not for pictures in the visual word form area. *Neuroimage*, 55(2):742–749, 2011.

[81] José Manuel Pena, Jose Antonio Lozano, and Pedro Larranaga. An empirical comparison of four initialization methods for the k-means algorithm. *Pattern Recognition Letters*, 20(10):1027–1040, 1999.

[82] Marc Proesmans and Luc Van Gool. Grouping based on coupled diffusion maps. In *Shape, Contour and Grouping in Computer Vision*, pages 196–213. Springer, 1999.

[83] Richard J Prokop and Anthony P Reeves. A survey of moment-based techniques for unoccluded object representation and recognition. *Graphical Models and Image Processing*, 54(5):438–460, 1992.

[84] Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Tracking human pose by tracking symmetric parts. In *Conference on Computer Vision and Pattern Recognition*, pages 3728–3735. IEEE, 2013.

[85] Daniel Reisfeld, Haim Wolfson, and Yehezkel Yeshurun. Context-free attentional operators: the generalized symmetry transform. *International Journal of Computer Vision*, 14(2):119–130, 1995.

[86] Andrew Richardson and Edwin Olson. Learning convolutional filters for interest point detection. In *International Conference on Robotics and Automation*, pages 631–637. IEEE, 2013.

[87] Edward Rosten and Tom Drummond. Machine learning for high-speed corner detection. In *European Conference on Computer Vision*, pages 430–443. Springer, 2006.

[88] Ethan Rublee, Vincent Rabaud, Kurt Konolige, and Gary Bradski. Orb: an efficient alternative to sift or surf. In *International Conference on Computer Vision*, pages 2564–2571. IEEE, 2011.

[89] Samuele Salti, Alessandro Lanza, and Luigi Di Stefano. Keypoints from symmetries by wave propagation. In *Conference on Computer Vision and Pattern Recognition*, pages 2898–2905. IEEE, 2013.

[90] Michaël Sassi, Maarten Demeyer, and Johan Wagemans. Peripheral contour grouping and saccade targeting: the role of mirror symmetry. *Symmetry*, 6(1):1–22, 2014.

[91] Gal Sela and Martin D Levine. Real-time attention for robotic vision. *Real-Time Imaging*, 3(3):173–194, 1997.

[92] Dinggang Shen, Horace HS Ip, Kent KT Cheung, and Eam Khwang Teoh. Symmetry detection by generalized complex (gc) moments: a close-form solution. *Transactions on Pattern Analysis and Machine Intelligence*, 21(5):466–476, 1999.

[93] Wei Shen, Kai Zhao, Yuan Jiang, Yan Wang, Xiang Bai, and Alan Yuille. Deepskeleton: Learning multi-task scale-associated deep side outputs for object skeleton extraction in natural images. *arXiv preprint arXiv:1609.03659*, 2016.

[94] Stephen M Smith and J Michael Brady. SusanâĂŤa new approach to low level image processing. *International Journal of Computer Vision*, 23(1):45–78, 1997.

[95] Michael Reed Teague. Image analysis via the general theory of moments. *JOSA*, 70(8):920–930, 1980.

[96] Ching L Teo, Cornelia Fermuller, and Yiannis Aloimonos. Detection and segmentation of 2d curved reflection symmetric structures. In *International Conference on Computer Vision*, pages 1644–1652, 2015.

[97] Matthias Sebastian Treder. Behind the looking-glass: A review on human symmetry perception. *Symmetry*, 2(3):1510–1543, 2010.

[98] Stavros Tsogkas and Iasonas Kokkinos. Learning-based symmetry detection in natural images. In *European Conference on Computer Vision*, pages 41–54. Springer, 2012.

[99] Tinne Tuytelaars and Krystian Mikolajczyk. Local invariant feature detectors: a survey. *Foundations and Trends in Computer Graphics and Vision*, 3(3):177–280, 2008.

[100] Olga Veksler, Yuri Boykov, and Paria Mehrani. Superpixels and supervoxels in an energy optimization framework. In *European Conference on Computer Vision*, pages 211–224. Springer, 2010.

[101] Yannick Verdie, Kwang Moo Yi, Pascal Fua, and Vincent Lepetit. Tilde: A temporally invariant learned detector. In *Conference on Computer Vision and Pattern Recognition*, pages 5279–5288. IEEE, 2015.

[102] Luc Vincent and Pierre Soille. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *Transactions on Pattern Analysis and Machine Intelligence*, 13(6):583–598, 1991.

[103] Paul Viola and Michael Jones. Robust real-time object detection. *International Journal of Computer Vision*, 4:34–47, 2001.

[104] Zhaozhong Wang, Lianrui Fu, and YF Li. Unified detection of skewed rotation, reflection and translation symmetries from affine invariant contour features. *Pattern Recognition*, 47(4):1764–1776, 2014.

[105] Zhaozhong Wang, Zesheng Tang, and Xiao Zhang. Reflection symmetry detection using locally affine invariant edge correspondence. *Transactions on Image Processing*, 24(4):1297–1301, 2015.

[106] Hermann Weyl. Symmetry princeton university press. *Princeton, NJ*, 1952.

[107] Nicolas Widynski, Antoine Moevus, and Max Mignotte. Local symmetry detection in natural images using a particle filtering approach. *Transactions on Image Processing*, 23(12):5309–5322, 2014.

[108] Bin Xiao, Gang Lu, Tong Zhao, and Liang Xie. Rotation, scaling and translation invariant texture recognition by bessel-fourier moments. *Pattern Recognition and Image Analysis*, 26(2):302–308, 2016.

[109] Bin Xiao, Jian-Feng Ma, and Xuan Wang. Image analysis by bessel–fourier moments. *Pattern Recognition*, 43(8):2620–2629, 2010.

[110] Pew-Thian Yap, Raveendran Paramesran, and Seng-Huat Ong. Image analysis by krawtchouk moments. *Transactions on Image Processing*, 12(11):1367–1377, 2003.

[111] Eiji Yodogawa. Symmetropy, an entropy-like measure of visual symmetry. *Attention, Perception, & Psychophysics*, 32(3):230–240, 1982.

[112] Hagit Zabrodsky and Daniel Algom. Continuous symmetry: A model for human figural perception. *Spatial Vision*, 8(4):455–467, 1994.

[113] Hagit Zabrodsky, Shmuel Peleg, and David Avnir. Hierarchical symmetry. In *International Conference on Pattern Recognition*, pages 9–12. IEEE, 1992.

[114] Hagit Zabrodsky, Shmuel Peleg, and David Avnir. A measure of symmetry based on shape similarity. In *Conference on Computer Vision and Pattern Recognition*, pages 703–706. IEEE, 1992.

[115] Zheng Zhang, Wei Shen, Cong Yao, and Xiang Bai. Symmetry-based text line detection in natural scenes. In *Conference on Computer Vision and Pattern Recognition*, pages 2558–2567, 2015.

[116] Liang Zheng, Yi Yang, and Qi Tian. Sift meets cnn: a decade survey of instance retrieval. *Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[117] Jian Zhou, Huazhong Shu, Hongqing Zhu, Christine Toumoulin, and Limin Luo. Image analysis by discrete orthogonal hahn moments. In *Image Analysis and Recognition*, pages 524–531. Springer, 2005.

[118] C Lawrence Zitnick and Krishnan Ramnath. Edge foci interest points. In *International Conference on Computer Vision*, pages 359–366. IEEE, 2011.